Heuristic Methods for Support Vector Machines with Applications to Drug Discovery

Robert Burbidge
University College London

Submitted in partial fulfilment of the requirements for the degree of PhD

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

The aim is not to finish the job quickly but to reach perfection.

'SOLOGDIN' IN THE FIRST CIRCLE BY ALEXANDER SOLZHENITSYN

Imperfection is an integral part of perfection.

IVOR CUTLER

Heuristic Methods for Support Vector Machines with Applications to Drug Discovery

by

Robert Burbidge

Abstract

The contributions to computer science presented in this thesis were inspired by the analysis of the data generated in the early stages of drug discovery. These data sets are generated by screening compounds against various biological receptors. This gives a first indication of biological activity. To avoid screening inactive compounds, decision rules for selecting compounds are required. Such a decision rule is a mapping from a compound representation to an estimated activity. Hand-coding such rules is time-consuming, expensive and subjective. An alternative is to learn these rules from the available data. This is difficult since the compounds may be characterized by tens to thousands of physical, chemical, and structural descriptors and it is not known which are most relevant to the prediction of biological activity. Further, the activity measurements are noisy, so the data can be misleading.

The support vector machine (SVM) is a statistically well-founded learning machine that is not adversely affected by high-dimensional representations and is robust with respect to measurement inaccuracies. It thus appears to be ideally suited to the analysis of screening data. The novel application of the SVM to this domain highlights some shortcomings with the vanilla SVM. Three heuristics are developed to overcome these deficiencies: a stopping criterion, HERMES, that allows good solutions to be found in less time; an automated method, LAIKA, for tuning the Gaussian kernel SVM; and, an algorithm, STAR, that outputs a more compact solution.

These heuristics achieve their aims on public domain data and are broadly successful when applied to the drug discovery data. The heuristics and associated data analysis are thus of benefit to both pharmacology and computer science.

Acknowledgements

This research was undertaken whilst I was an associate of the Postgraduate Training Partnership established between Sira Ltd and University College London. Postgraduate Training Partnerships were a joint initiative of the UK Department of Trade and Industry (DTI) and the Engineering and Physical Sciences Research Council (EPSRC). Support by the DTI and EPSRC is gratefully acknowledged.

The research was also undertaken within the INTErSECT Faraday Partnership managed by Sira Ltd and the National Physical Laboratory, and has been supported by EPSRC, GlaxoSmithKline, and Sira. The data mining package Clementine was provided by SPSS.

Research is dependent upon personal contributions and support as much as it is upon financial ones. The following is a probably incomplete list of the individuals who have either actively or passively guided, aided, nudged, cajoled, or inspired me during the production of this thesis.

My principal supervisor, Bernard Buxton at UCL, provided guidance and background in areas with which I was not familiar. The first being computer science, and the second being research. Bernard provided me with open-minded encouragement as I explored many paths.

I owe a great deal of gratitude to Steven Barrett of GSK. Steven did a lot of data processing and analysis to ensure that the data with which I was provided was in a form that was amenable to analysis. Steven also provided me a wealth of literature in numerous areas of pharmacology and data mining.

I offer many thanks to Bernard and Steven for their painstaking reading and rereading of my various internal reports, external publications and especially this thesis itself. Any errors, whether orthographical or technical, remain mine.

The development of ideas, many abandoned, some included, was assisted freely by my supervisor at Sira, Mark Hodgetts. Mark also read and commented on much of my early work. Thanks are due also to John Gilby of Sira for general guidance and encouragement, particularly with respect to doing research work within an industrial setting.

There are many others at UCL who were of help. The research assistants of the INTErSECT project, Apollo Tankeh and Bill Langdon. My second supervisor Sean Holden. Denise Gorse for challenging and constructive criticism on my transfer from MPhil to Phd. Matthew Trotter, David Corney, Martin Sewell, Dirk Weirich, Tim Gordon, Gordon Hunter, Allan Erskine, Peter Bentley, and numerous other individuals provided edifying, stimulating and occasionally visionary discussions on a swathe of topics from the practicalities of implementation to the philosophical validity of machine learning as a discipline.

I thank Dave Denison of Imperial College for providing me with a not entirely unrelated research position. This enabled me to continue working with SVMs and related techniques and gave me an alternative viewpoint.

I would like to thank those who have put up with me during the seemingly interminable process of researching and writing this thesis: my parents, Christine and David; my siblings, Karen and

Chris; Katie, Jules, and Emma; Ravi, Phil, Helen, Rosie and Mr Singh; Joshua, Collette and Dipak; and Sammy Hilbert-Spaess.

Contents

1 Introduction **16**

1.1 Supervised Learning 16
1.2 Drug Discovery 20
1.3 The Idea of Structural Risk Minimization 22
1.4 Heuristics for Support Vector Classification 24
1.5 Synopsis 26

2 Drug Discovery **31**

2.1 The Drug Discovery Process 32
2.1.1 High Throughput Screening 33
2.1.2 Screening Set Selection 35
2.1.3 Library Design 35
2.1.3.1 Diverse Libraries 35
2.1.3.2 Lead Libraries 36
2.1.3.3 Combinatorial Libraries 36
2.1.3.4 Optimizing ADME properties 36
2.2 Representing Compounds 37
2.2.1 Identifying Descriptive Features 37
2.2.2 Structural Keys 38
2.2.3 Fingerprints 39
2.2.4 Affinity Fingerprinting 39
2.3 Review of Current Approaches 40
2.3.1 QSAR — Rational Design 40
2.3.2 Similarity Measures 42
2.3.3 Clustering 43
2.3.4 Decision Trees 44
2.3.5 Neural Networks 44
2.3.5.1 Multi-Layer Perceptrons 45
2.3.5.2 Radial Basis Function Networks 45
2.4 Pharmaceutical Data Analysis 46
2.4.1 Predicting the Inhibition of Dihydrofolate Reductase 46
2.4.1.1 Algorithms 47
2.4.1.2 Results and Discussion (QSAR) 48
2.4.2 Analysis of HTS Data 49
2.4.2.1 Measuring Performance 49
2.4.2.2 Results and Discussion (HTS) 50

2.5 Summary . . . 51

3 Support Vector Classification **55**
3.1 The Support Vector Machine . . . 56
3.1.1 Binary Classification . . . 56
3.1.2 Margin Maximization as a Quadratic Program . . . 57
3.1.3 Implementation . . . 59
3.1.4 Variable Misclassification Costs . . . 61
3.1.5 Bounds on the Expected Error . . . 62
3.1.5.1 Bounds from VC-Theory . . . 62
3.1.5.2 Bounds on the Leave-One-Out Error . . . 63
3.1.5.3 Empirical Comparison . . . 65
3.2 Performance Measures . . . 70
3.2.1 Receiver Operating Characteristic Curves . . . 70
3.2.2 Relative Advantage . . . 72
3.3 Applications . . . 73
3.3.1 Predicting the Inhibition of Dihydrofolate Reductase . . . 73
3.3.2 Predicting Activity of Competitive Antagonists . . . 74
3.3.2.1 Competitive Antagonists . . . 74
3.3.2.2 Results and Discussion (SVC pK_i) . . . 76
3.3.3 Analysis of HTS Data . . . 77
3.3.3.1 Ranking Compounds . . . 78
3.3.3.2 Results and Discussion (SVC HTS) . . . 79
3.4 Summary . . . 80

4 Heuristics for SVC **83**
4.1 Stopping Criteria . . . 83
4.1.1 Persistence . . . 84
4.1.2 Results and Discussion (HERMES) . . . 85
4.2 Automated Model Order Selection . . . 87
4.2.1 Incremental Tuning . . . 88
4.2.2 Heuristic Approaches . . . 89
4.2.3 Approximating the Support Vector Set . . . 90
4.2.4 Reducing Training Time . . . 92
4.2.5 Stopping Criterion . . . 94
4.2.6 Results and Discussion (LAIKA) . . . 94
4.3 Inducing Sparsity . . . 97
4.3.1 Data Cleaning . . . 97
4.3.2 Results and Discussion (STAR) . . . 98
4.4 Summary . . . 99

5 Application to pK_i Data **103**
5.1 Some Data Mining Considerations . . . 104
5.1.1 Training Data Selection . . . 104
5.1.1.1 Performance Comparison . . . 105
5.1.1.2 Results and Discussion . . . 106
5.1.2 Misclassification Costs . . . 108

5.1.2.1 Random vs Centroids . . . 109
5.1.3 Meta-Analysis . . . 109
5.1.3.1 Relationship Between Performance Measures . . . 109
5.1.3.2 Behaviour of Error Estimators . . . 110
5.1.4 Summary of Data Mining Considerations . . . 111
5.2 Evaluation of Heuristics . . . 112
5.2.1 STAR . . . 112
5.2.2 LAIKA . . . 112
5.2.3 Summary of Heuristics . . . 115
5.3 Alternative Compound Representations . . . 115
5.3.1 Predicting from Structural Keys . . . 115
5.3.1.1 Clustering Structural Keys . . . 118
5.3.1.2 STAR on Structural Keys . . . 119
5.3.2 Predicting from Reduced Graphs . . . 119
5.3.3 Predicting from Bioactivity . . . 120
5.3.3.1 STAR on Affinity Fingerprints . . . 122
5.3.3.2 LAIKA on Affinity Fingerprints . . . 123
5.3.3.3 STAR+LAIKA on Affinity Fingerprints . . . 123
5.3.4 Summary of Alternative Representations . . . 124
5.4 Summary . . . 126

6 Critical Assessment, Further Work, and Conclusions 129
6.1 Critical Assessment . . . 130
6.1.1 General Criticisms . . . 131
6.1.2 QSAR . . . 132
6.1.3 HTS . . . 134
6.1.4 pK_i . . . 135
6.1.5 HERMES . . . 137
6.1.6 LAIKA . . . 138
6.1.7 STAR . . . 139
6.2 Further Work . . . 139
6.2.1 General Comments . . . 140
6.2.2 Specific Ideas . . . 141
6.2.2.1 QSAR . . . 141
6.2.2.2 HTS . . . 141
6.2.2.3 pK_i . . . 142
6.2.3 New Directions . . . 142
6.2.3.1 Transduction . . . 142
6.2.3.2 Imputation of Missing Data . . . 143
6.2.3.3 Maximixing Relative Advantage . . . 144
6.2.3.4 Aggregating SVMs . . . 144
6.3 Conclusions . . . 144

List of Figures

3.1 Regularization: Diabetes. The test error (*solid line*) has a minimum at around $C = 1$. The proportion of training examples that become support vectors (*dotted line*) decreases as C increases. When C is too small the SVM underfits the training data, when C is too large then the SVM overfits the training data. The SVM was trained with a Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-\|x - z\|^2/d)$. Results are averaged over ten partitions of the data into training and test sets of equal sizes. 60

3.2 Error Bound Comparison: Cancer, $C = 1$. The minimizer, $\hat{\sigma}$, of the $\alpha\xi$ estimate (*dotted line*) underestimates the minimizer, σ^*, of the test error (*solid line*). The test error obtained, $\varepsilon(\hat{\sigma})$, is close to the optimal, $\varepsilon(\sigma^*)$. 66

3.3 Error Bound Comparison: Diabetes, $C = 1$. The test error (*solid line*) attains a minimum at around $\sigma = 1.5$ and increases slightly afterwards. The minimizer of the $\alpha\xi$ estimate (*dotted line*) underestimates the minimizer of the test error. 66

3.4 Error Bound Comparison: HTS, $C = 1$. The test error (*solid line*) is more-or-less constant over the range of σ tried. The $\alpha\xi$ estimate (*dotted line*) increases with σ. The behaviour of the $\alpha\xi$ estimate is qualitatively different from that of the test error on these data. 67

3.5 Risk Bound Comparison: Cancer, $C = 1$. The minimizer of the $\alpha\xi$ estimate (*dotted line*) is close to the minimizer of the test risk (*solid line*). The relative advantage (*dash-dotted line*) varies inversely with the test risk. 67

3.6 Risk Bound Comparison: Diabetes, $C = 1$. The minimizer of the $\alpha\xi$ estimate (*dotted line*) is the minimizer of the test risk (*solid line*). The solution is unstable for $\log(\sigma) > 0.5$ so the error estimate does not hold. The relative advantage (*dash-dotted line*) varies inversely with the test risk. 69

3.7 Risk Bound Comparison: HTS, $C = 1$. The $\alpha\xi$ estimate (*dotted line*) is tighter when the risk (*solid line*) is 1. This corresponds to predicting all positives or all negatives. The $\alpha\xi$ estimate is qualitatively the same for the other values, although overall it is not predictive since it is too loose. The relative advantage (*dash-dotted line*) varies inversely with the test risk. 69

3.8 ROC curves: Cancer. The ROC curve is generated by varying the decision threshold b for an SVM ($C = 1$). Neither the RBF kernel (*dashed line*) nor the linear kernel (*dotted line*) has an ROC entirely containing the other. Both are much better than random (*solid line*). Note that only a small portion of the ROC curves are shown, as beyond a false positive rate of 0.06 both classifiers have a true positive rate of 1. 71

4.1 HERMES. An SVM with Gaussian kernel was trained on 50% of the data, with $C = 1000$. The $\alpha\xi$ estimator (*dotted line*) is qualitatively the same as the test error (*solid line*) during the later stages of training. The behaviour of the $\alpha\xi$ estimator can thus be used to define a heuristic stopping criterion. Note that the test error attains its minimum after only a few iterations. Moreover, the test error is rather poor, both for the Diabetes data (*left*) and the Ionosphere data (*right*), thus stopping early is advisable when searching for a good value of C or σ. 85

4.2 LAIKA: Cancer $C = 1$. Training time decreases as h_a increases, since fewer updates are being made. Versions 2.1 (*crossed, dashed line*) and 2.2 (*crossed, dotted line*) are fastest. Versions 1.1 (*circled, dashed line* and 1.2 (*circled, dotted line*) approach the fastest as h_a increases. Version 2.0 (*crossed, solid line*) is slowest. 93

4.3 LAIKA: Diabetes $C = 1$. Training time decreases as h_a increases, since fewer updates are being made. Versions 1.1 (*circled, dashed line*) and 2.1 (*crossed, dashed line*) are fastest. Versions 1.2 (*circled, dotted line*), 2.0 (*crossed, solid line*) and 2.2 (*crossed, dotted line*) approach the fastest as h_a increases. 93

4.4 HERMES and LAIKA. As in Figure 4.1, the $\alpha\xi$ estimator (*dotted line*) is qualitatively the same as the test error (*solid line*) during training. Note that the test error is much lower on Diabetes when using LAIKA than using the original value (cf. Figure 4.1). 94

5.1 pK_i. The error (*left*) of the SVM ($C = 10$) when training on the centroids is higher on five assays. However, the *AUROC* (*right*) is approximately the same or higher on ten assays, which suggests the operating point has not been set optimally. . . . 106

5.2 pK_i. The relative differences between the (test) population priors, π_{+1}, and the (training) sample priors, π^s_{+1}, are larger when the sample data are selected by clustering. This leads to the operating point being set sub-optimally (cf. Figure 5.1). . 107

5.3 pK_i. When using weighted losses the risk is reduced by training on the centroids. . 109

5.4 pK_i. STAR outputs a solution with fewer support vectors (N_{SV}) than the SVM both for $C = 1$ (*left*) and $C = 10$ (*right*). The training time required by STAR is in general lower than for the SVM. 113

5.5 pK_i. The *AUROC* obtained by the new version of LAIKA is always lower when $C = 1$ (*left*) and approximately the same when $C = 10$ (*right*). 113

5.6 pK_i. The final value of σ (*left*) for LAIKA v2.2 when $C = 1$ is much larger than the estimate σ_{JDH}. This led to a high training error (underfitting). When $C = 10$, the final value of σ was about 50% higher, which resulted in similar predictive performance with fewer SVs (*right*). 114

5.7 pK_i. Comparison of SVM performance when using the properties ('Props') or MACCS keys ($C = 1$): test error (*top left*), *RA* (*top right*), *AUROC* (*bottom left*). Note that $\varepsilon^l_{\alpha\xi}$ (*bottom right*) correctly predicts that the structural keys will give better performance an all assays. 117

5.8 pK_i. When training on the MACCS keys, selecting the training data by clustering on fingerprints increases the *AUROC* on six assays, but this does not correspond to an increase in accuracy (*left*). The training time and number of SVs were increased (*right*). 118

5.9 pK_i. When using structural keys as descriptors, STAR reduces the number of SVs by roughly one third to one half. The cost incurred is no more than a 40% increase in runtime. 119
5.10 pK_i. The test error (*left*) when using affinity fingerprints ('pK_i') is lower than when using the properties ('Props') or reduced graphs ('RG'). The training time is also lower (*right*). (The test error for the MACCS keys was the same as for the reduced graphs; the training time was an order of magnitude longer.) Results are shown for $C = 1$. 122
5.11 pK_i. STAR reduces the number of SVs (N_{SV}) at the expense of longer runtime. The relative gain compared to the increased time is larger for $C = 1$ (*left*) than for $C = 10$ (*right*) 123
5.12 pK_i. When $C = 10$ LAIKA increases the estimate of σ over its starting value (σ_0), leading to a reduction in the number of SVs (N_{SV}) 124

List of Tables

1.1 Types of attributes, with example instantiations. 17

2.1 The traditional path of rational drug discovery. 33
2.2 A 'reverse pharmacology' facilitated by advances in HTS and genomics. 34
2.3 Predicting the inhibition of dihydrofolate reductase: Estimated generalization errors and training times averaged over the five cross-validation folds. The manually tuned MLP has the best performance, but the longest training times. The RBF performed badly. (Reproduced from Burbidge et al. (2000).) 48
2.4 Predicting the inhibition of dihydrofolate reductase: Model selection time for the neural networks. No model selection was performed for C5.0. The manually tuned neural network required the most computation time, and this led to the best performance (see Table 2.3). (Reproduced from Burbidge et al. (2000).) 48
2.5 HTS Statistics. The number of compounds screened on each assay (for which features were available) is shown with the hit rate π^s_{+1} (the proportion of positive examples in the training sample) for each assay. 50
2.6 HTS Results: fs01. The highest global enhancement for each assay is emphasized. Logistic regression wins five out seven times. Across these assays, the neural networks are not better than random selection on average and never win in comparison to the linear techniques. 52
2.7 HTS Results: fs02. The highest global enhancement for each assay is emphasized. The linear techniques win six times out of seven. For the linear techniques, classification and regression have approximately equal performance. This also holds to a slightly lesser extent for the neural networks. Across these assays, the neural networks are not better than random selection on average. 52
2.8 HTS Results: fs03. The highest global enhancement for each assay is emphasized. Logistic regression wins four out of seven times. Across these assays, the neural networks are not better than random selection on average and again never win. . . 52
2.9 HTS Results: Summary. The highest global enhancement (GE) for each assay is shown with the corresponding algorithm and feature set. The linear techniques win on all of the assays. Logistic regression wins four out of seven times and is tied on a8. 52

3.1 Competitive Antagonists: Proportion of active compounds (positives) in each of the training sets. Since these compounds are further downstream in the drug discovery process than those screened in HTS there is a higher proportion of actives to inactives, between 2:5 and 7:3. 75

3.2 Competitive Antagonists. Error rate for SVM, with a Gaussian kernel. The test error is much higher than the training error, i.e. the SVM is overfitting the data (t and p values are shown for the null hypothesis that the SVM is not overfitting). The last two columns show the area under the ROC curve ($AUROC$) and the relative advantage (RA). C^* is that value minimizing the $\alpha\xi$ estimator. 76
3.3 Competitive Antagonists. Test error rates for C5.0 and a dynamic neural network. The performance of C5.0 is comparable to that of the Gaussian SVM. The dynamic neural network performs badly. t and p values are shown for the one-tailed test that C5.0 or Dynamic has the same performance as the SVM. 77
3.4 Competitive Antagonists. Number of iterations (N_{iter}) and number of SVs (N_{SV}) for the standard SVM. Training requires more iterations for $C = 10$ since the search space for the Lagrange multipliers is larger ($0 \leq \alpha_i \leq C$). Fewer SVs are needed to construct the decision surface at $C = 10$ as the function class is larger. 78
3.5 HTS Results: LOOMS. The global enhancement (GE) over random screening is shown for the LOOMS software (*left*). This performs automated model order selection for an SVM with Gaussian kernel. The best performance (*emphasized*) is the same as the best classical technique (logistic regression, *right*) on one assay, poorer on four assays and better on two. The range of GE is surprisingly large. Although assay a6 with feature set fs01 was easiest for the classical techniques, the GE for the SVM is over three times higher. 79
3.6 HTS Results. The global enhancement (GE) over random screening is shown for the SVM when trained with unequal misclassification costs. The best performance (*emphasized*) is better than LOOMS on five assays, and poorer on two assays. . . . 80

4.1 HERMES. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$, and persistence of 100. Individually, the results are not significantly different, at 95%, from the standard SVM. 86
4.2 BSVM. 10 fold CV error rates (Hsu and Lin, 2002). The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$. Individually, the results are not significantly different, at 95%, from using HERMES. 86
4.3 HERMES. $C = 1$. Since the number of iterations required is fairly small when C is small, the SVM converged before the HERMES criterion was satisfied. The minor differences in the results are due to the differing implementations of the QP solver. 87
4.4 HERMES. $C = 1000$. The SVM was halted by the HERMES criterion on all five data sets. The reduction in the number of iterations (N_{iter}) required varies from 43%–77%. The price paid is a slight increase in the number of support vectors (N_{SV}) and the number of training errors (N_{err}). 87
4.5 LAIKA Version 1: Cancer, $C = 1$. The test error is almost constant for different values of the update frequency h_a. Versions 1.1 and 1.2 slightly outperform version 1.0. The errors are lower than for $C = 10$. 91
4.6 LAIKA Version 1: Cancer, $C = 10$. The test error is almost constant for different values of the update frequency h_a. Versions 1.1 and 1.2 slightly outperform version 1.0. The errors are higher than for $C = 1$. 91
4.7 LAIKA Version 1: Diabetes, $C = 1$. The test error is almost constant for different values of the update frequency h_a. Versions 1.2 slightly outperforms version 1.1. The errors are lower than for $C = 10$. 92

4.8 LAIKA Version 1: Diabetes, $C = 10$. The test error is almost constant for different values of the update frequency h_a. Versions 1.2 slightly outperforms version 1.1. The errors are higher than for $C = 1$. 92
4.9 SVM. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with width σ_{JDH}. That minimum error over the three values of C is emphasized. 95
4.10 LAIKA. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with the width parameter updated by LAIKA version 2.2. That minimum error over the three values of C is emphasized. 95
4.11 SVM. Training times (in minutes). The SVM was trained with a Gaussian kernel, with width σ_{JDH}. 96
4.12 LAIKA. Training times (in minutes). The SVM was trained with a Gaussian kernel, with the width parameter updated by LAIKA version 2.2. 96
4.13 LAIKA. Percentage reduction in number of SVs. 96
4.14 STAR. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$, and $h_e = 100$. Error rates significantly different, at 95%, from the standard SVM are emphasized (cf. Table 4.1). The last row shows the p-values for the two-tailed t-test of the null hypothesis that STAR and the SVM have the same error rate over the five data sets. 98
4.15 STAR. The number of SVs (N_{SV}) in the solutions when using STAR is never more than that of using the standard SVM and was up to 84% lower. The number of iterations (N_{iter}) was in general increased. 99

5.1 pK_i. Training the SVM with weighted misclassification costs leads to a slight decrease in relative advantage (RA) and a slight increase in the number of support vectors (N_{SV}). The number of iterations was decreased when $C = 1$ and increased when $C = 10$. $AUROC$ is almost unchanged. 108
5.2 pK_i. Correlations between performance measures, when minimizing the specified loss functional. 110
5.3 pK_i. Test error rates for the SVM when using the properties ('Props') or MACCS keys. The error on the MACCS keys is significantly lower on nine assays (t-test at 95%). 116
5.4 Rescaling of pK_I values to a semi-quantitative (SQ) ranking. 121
5.5 pK_i. Test error rates for the SVM when using the reduced graphs ('RG') or affinity fingerprints ('pK_i'). The error on the affinity fingerprints is significantly lower on ten assays (t-test at 95%). Results are shown for $C = 1$. 121
5.6 pK_i.Summary of SVM training and solutions for the four compound representations. 125

6.1 Method-to-criterion. A brief summary of my learnings for three algorithms, when applied to pharmaceutical data analysis. 130

Notation

The following notation is used in this thesis, unless stated otherwise.

$\Re$	the set of reals
$\mathcal{X}$	an abstract domain
ln	logarithm to base e
log	logarithm to base 10
$\langle \mathbf{x}, \mathbf{z} \rangle$	inner product between two vectors $\mathbf{x}$ and $\mathbf{z}$
$\|\cdot\|$	2-norm (Euclidean distance) $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
$L^2(\mathcal{X})$	space of functions on $\mathcal{X}$ square integrable w.r.t. Lebesgue measure
$E(\cdot)$	expectation operator
$\Pr\{\cdot\}$	probability of an event
d	dimensionality of input space
l	number of training examples
$\mathbf{x}_i$	input patterns
y_i	target values or classes
y	vector of target values
$\mathbf{w}$	weight vector
b	threshold (or operating point)
f	a decision function $f : \Re^d \to \{-1, +1\}$
$\mathcal{F}$	a family of decision functions f
$\lambda(l, k)$	cost of misclassifying an element of class k as an element of class l
$R(f)$	risk of f
R_{emp}	empirical risk of f
K	Mercer kernel
σ	'width' parameter of an RBF kernel
α_i	Lagrange multiplier
ξ_i	slack variable (margin error)
$\varepsilon^l_{\alpha\xi}$	$\alpha\xi$ estimator of generalization error
π^s_y, π_y	sample, true proportion of class $y \in \{-1, +1\}$
C, C^+, C^-	regularization parameters
Q	Hessian of a quadratic program
ϵ	termination precision
ε	generalization error
k	norm on errors

Chapter 1

Introduction

> *Don't estimate the required capacity of a bridge by the number of people who swim across the river today.*
>
> ANON

The work presented in this thesis has been motivated by the problems arising in the analysis of pharmaceutical data sets. The main tool used to analyse the data, and to make predictions, is the recently developed support vector machine (SVM). During the analysis of the pharmaceutical data, some shortcomings of the SVM are identified. Heuristics are proposed to overcome these. These new tools are then applied to the problems arising in the early stages of drug discovery.

1.1 Supervised Learning

Without thinking about it, human beings continually process a mass of sensory data into distinct objects and then put these objects into categories. Anyone of good hearing can rapidly process a mass of auditory data and identify various sounds, such as 'motorcycle engine', 'classical music', etc. A mechanic may be able to further subdivide the former category into 'functioning normally' and 'malfunctioning'. A musician may be able to subdivide the latter into 'Sibelius', 'Mahler', 'Ravel', and so on. Although we have only a basic understanding of how the brain performs this remarkable task, functionally, it can be split into two stages: object recognition and object classification. This is best illustrated by considering the sense of sight. Light reflected from various objects is incident upon the rods and cones at the back of the eyes. Thus stimulated, these then convey electrical signals to the brain. At this stage the visual cortex has a mass of structured data, there are no 'objects' in the mind as yet. The first stage is to segment this image into 'objects'. Once these 'objects' have been recognized they are then labelled. (The relationship between these labelled 'objects' in the mind and the objects which led to them is a matter for philosophers.) For example, a chunk of data that originated in the lower left portion of the visual field may be recognized as an 'object' and then labelled as 'book'. This labelling of objects into pre-defined categories will be called *classification*. (The term *classification* is not used here in the sense of defining the categories themselves, but the act of placing objects into pre-defined classes.) There is also the possibility of 'don't know' or 'novelty'. A 'don't know' labelling may occur if, for example, the musician above was asked to classify a motorcycle engine as 'good' or 'bad' — he may not have enough experience of motorcycles to distinguish the two. A 'novelty' labelling may occur if the musician was played a piece by an obscure composer he had never heard before.

The key reason that the classifications in the above examples did not fall into the pre-defined

Type	Object	Attribute	Labels
nominal	symphony	composer	'Sibelius', 'Mahler', 'Ravel'
binary	bit	state	'0','1'
discrete	shoe	size	$3, 3\frac{1}{2}, 4$
continuous	people	height	$\Re^+$

Table 1.1: Types of attributes, with example instantiations.

categories (that is, 'good', 'bad'; 'Sibelius', 'Mahler', 'Ravel') was lack of experience on the part of the musician. Contrapositively, if classification is to be performed successfully, experience is required. That is, in order to classify objects one must have *learned* what the classes *are* and one must have *learned* a *decision strategy* for placing objects into classes. The problems presented in this thesis are of the latter type, that is *supervised classification.* The learning is termed 'supervised' since in order to assign an object to a class, e.g. 'Sibelius symphony', one must have been told by a supervisor, or teacher, which objects are in that class.

In reality, things aren't always as clear cut as described above. A car engine may be 'functioning perfectly', 'good enough for domestic use', 'in need of maintenance', 'not working', etc. A book may be 'red' or 'green' or a shade in between. A measurable attribute associated with a set of objects may be: nominal, binary, discrete, or continuous. An example of each type is given in Table 1.1. The binary case can be thought of as a special case of the nominal case, or of the discrete case. It is listed separately since it is often useful to treat the other cases as one or more binary problems. For example, although there is an infinite number of colours between red and green, it is sometimes useful to have a boundary point. This would then allow simple classification of objects as 'red' or 'green'. Since colours near the boundary point will be mostly yellow, the exact boundary point will vary from person to person.

The problem of labelling a set of objects in terms of an ordered discrete set of labels, i.e. ranking them according to the level of some attribute, is termed *ordinal regression.* The problem of assigning labels to objects from a continuous set of labels is known as *regression.*

A decision strategy for classification, ordinal regression or regression is a mapping

$$\begin{aligned} X &\rightarrow Y, \\ x &\mapsto y, \end{aligned}$$

which maps objects x in some set X to labels y in a set of labels Y. Human beings are generally quite good as this, particularly when the objects x are from their field of expertise. Human beings cannot, however, label the very high numbers of objects that businesses and research institutions handle on a daily basis. For example, given a set of 100 organic compounds (molecules), a trained medicinal chemist may be able to label them as 'inert', 'toxic', 'potentially therapeutic', etc. However, there are potentially trillions of compounds in which a pharmaceutical company may be interested, so even 1000 medicinal chemists, examining 100 compounds a day each, would take hundreds of thousands of years to examine them all.

The human brain is a massively parallel computing system, which seems to make it well-suited for identifying objects and labelling them. However, signal transmission, and hence information processing, is fairly slow in the brain, in comparison to modern computers. Computers can process information much faster than the human brain, but cannot easily recognize and label objects. The task then, is to develop a decision strategy that gives close to human performance that can be implemented, rapidly, in a computer.

One possible approach to this problem is to attempt to encode in a computer the methodology of an expert. This is a very tricky problem though, as most experts find it difficult to verbalize exactly how they make their decisions. (How do you decide whether or not 'sky the blue is' is syntactically correct?) It is also very expensive to hire experts and skilled interrogators (known as knowledge engineers or elicitation experts). A more appealing alternative is to mimic the process by which the expert became an expert. As mentioned above, the human expert has learned a decision strategy based on past experience. That is, the expert has been trained. This training process essentially consists of seeing objects $\{x_1, x_2, \ldots, x_l\}$ and assigning labels $\{f(x_1), f(x_2), \ldots, f(x_l)\}$. A 'teacher' then supplies the correct labels (insofar as they can be correct) $\{y_1, y_2, \ldots, y_l\}$ and the trainee implicitly or explicitly adjusts her decision strategy f accordingly. The teacher may be a human being, a textbook, nature, experience, etc.

Formally, we desire a learning machine that, given a *training set*

$$\{x_1, \ldots, x_l\} \in X^l \tag{1.1}$$

with associated labels

$$\{y_1, \ldots, y_l\} \in Y^l, \tag{1.2}$$

produces a decision strategy, or *hypothesis*,

$$\begin{aligned} f : X &\rightarrow Y, \\ f : x &\mapsto f(x), \end{aligned}$$

such that $f(x)$ is close to y for the majority of objects from X that we are likely to encounter in the future. Note that we are not demanding that $f(x) = y$ for all $x \in X$, only that $f(x)$ is approximately equal to y most of the time. Demanding $f(x) = y$ for all $x \in X$ is usually too stringent a requirement. We may also desire that f has other properties that are important to the solution of the original problem. For example, that it is easy to compute in terms of memory and time requirements, or that it has a simple representation.

What 'close' means depends on the context. For classification we may desire that the *risk*

$$R(f) = \Pr\{x \in X | f(x) \neq y\}, \tag{1.3}$$

i.e. the probability of misclassifying an arbitrary object, is small.

In cancer diagnosis it is generally the case that misclassifying healthy cells as cancerous is not as serious as misclassifying cancerous cells as healthy. In the former case, unnecessary expense and stress to the patient will ensue in the form of further tests and treatments, which will eventually show that a mistake has been made. In the latter case, the patient may develop cancer, which is much more expensive to treat, is more stressful and is potentially fatal. In this case, we may desire that

$$\begin{aligned} R(f) &= \lambda(-1,+1) \Pr\{x \in X^{+1} | f(x) = -1\} \\ &+ \lambda(+1,-1) \Pr\{x \in X^{-1} | f(x) = +1\} \end{aligned}$$

is small, where $\lambda(-1,+1)$, the cost of misclassifying cancerous cells ($x \in X^{+1}$, $y = +1$) as healthy ($x \in X^{-1}$, $y = -1$), is higher than $\lambda(+1,-1)$, the cost of misclassifying healthy cells as cancerous. These costs may not be known beforehand, and may change with time and the opinions of different oncologists. Thus, it is often desirable to have a decision strategy that performs well over a range

of costs.

For regression, when the label is continuous, e.g. $y \in \Re$, we may desire that

$$R(f) = E(\|f(x) - y\|_p) \tag{1.4}$$

i.e. the expected deviation of the prediction from the truth, is small, where $\|\cdot\|_p$ is a norm on the set $Y = \Re$ given by

$$\|z\|_p = |z|^p \tag{1.5}$$

and E is the expectation operator.

Apart from the risk of using a decision strategy to label objects, there are a number of other considerations to be taken into account when constructing a learning machine. Which of these is most important depends upon the problem domain. Below I summarize some properties of a learning machine that have been suggested as performance criteria.[1]

Accuracy This is an immediately obvious requirement of any predictive algorithm: we wish to learn an hypothesis that generalizes to unseen examples.

Time The time available for learning an hypothesis is a key factor when constructing a learning machine. There is often a trade-off between accuracy and speed: a learning machine that outputs a less accurate decision strategy may be preferred over one outputting a more accurate one if the latter requires orders of magnitude more training time or prediction time.

Memory Learning machines that require the entire training set to be held in memory are not appropriate for some massive data sets that are being generated by business and research institutions. Memory requirements are also an issue in labelling, e.g. handwritten digit recognition for palm top computers.

Convergence A learning machine may be deterministic or stochastic. In the latter case it may be guaranteed to output a good decision strategy with a certain probability.

Interpretability Data analysis is often automated owing to the sheer size of the data sets involved. This can often lead to problems of interpretability. In some countries it is a legal requirement of a credit refusal to have a reason, not simply "the black box says you're a credit risk". Humans are understandably wary of black boxes, e.g. in medical decision making.

Robustness Many real-world data sets are corrupted by noise, either in the description of the objects or in their labels. How a learning machine is affected by this will determine its suitability in such domains.

Flexibility Is the learning machine to be used on one problem only, or a class of similar problems? In real world applications the objects of interest may vary in their form and in their descriptions.

Performance over a range of costs The misclassification costs may not be equal, may be variable or may not be known at all beforehand.

Data handling Objects may be described by various types of attributes, as listed in Table 1.1. Many learning machines, in particular those from statistics, are designed only to handle continuous attributes, which could be a shortcoming in some domains.

[1] With thanks to Peter Bentley.

Tunability This can be a boon or a burden. The ability to change the settings to suit the problem is desirable, but having free parameters means that they need to be set, which can take a considerable amount of time and human effort. Ideally, a learning machine should have an automated method to set its parameters, with user intervention allowed if desired.

In the next section the requirements of a learning machine for labelling compounds during the early stages of drug discovery are discussed. In Section 1.3 the support vector machine is introduced, and its suitability for the pharmaceutical problems posed with respect to the above performance criteria is discussed.

1.2 Drug Discovery

Supervised learning would be confined to the realms of theoretical computer science without applications. The application of supervised learning presented in this thesis is to the problem of drug discovery. Specifically, to the early stages of drug discovery where it is necessary to whittle down the the trillions of potential (i.e. as yet unsynthesized) therapeutic agents to a handful of compounds that can be investigated in detail by chemists, pharmacologists, biochemists and so on in order to develop a useful, marketable, effective drug.

In the early stages of drug discovery, pharmaceuticals companies physically screen tens to hundreds of thousands of compounds against therapeutic targets in order to identify those exhibiting desirable biological activity. This process, known as *high throughput screening* (HTS) has become possible due to advances in assay technology, robotics and information processing. A new field, *chemoinformatics*, has been developed to cope with the subsequent processing of the massive volume of data generated by HTS. Even though millions of compounds are screened annually, this is not sufficient to exhaust the trillions of possible 'drug-like' compounds. One goal, therefore, is to preferentially order compounds for screening. If a learning machine can be developed to order compounds such that those at the top of the list are more likely to exhibit appropriate biological activity then the number of active compounds, or *hits*, discovered by HTS will be increased. This is one way to avoid the problem of screening everything. This is an ordinal regression problem and as such can be tackled by classification or regression. The classification approach is to learn a decision strategy for identifying hits, that outputs a confidence in its labelling. The compounds can then be ranked in terms of 'confidence of hit'. The regression approach would be to learn a mapping from the compounds to their real-valued activity. As there is a huge volume of screening data, there are plenty of training data for learning such a decision strategy. By the same token, human analysis of the raw data is impossible, necessitating automated summarization and learning techniques.

Activity estimates from HTS are subject to variation due to experimental conditions and the small amounts of compounds used. Compounds determined as hits on an initial screen are subjected to further screens in order to obtain more accurate estimates of their activity. At each successive stage the number of compounds is reduced, and their bioactivities better characterized. Data from these later, more accurate, screens can be used to learn further decision strategies for compound selection and for prediction of activity. At each stage the data have particular characteristics which must be taken into account when learning each new decision strategy.

An absolutely vital aspect of supervised learning, which up to now has not been made explicit, is the description of the object to be labelled. In the case of drug discovery, we cannot physically input compounds into a computer. Instead, we must input a collection of numbers, perhaps with some structure, that describe that compound. For the purpose of labelling compounds as 'drug' or

'non-drug', these may be such things as: molecular weight, solubility, polarizability, etc. It is an open problem in most applications as to what description is optimal. This topic, in the context of drug discovery, is dealt with in more detail in Chapters 2 and 5, but remains, ultimately, insoluble. It may seem sensible to input all available information about an object in a learning machine, but this is frequently misleading. Certain attributes of the object may be irrelevant for classification. Some attributes may be poorly measured and do more harm than good, even if they are relevant.

In Section 1.1 above, some desirable properties of a learning machine were outlined. In the application of supervised learning to drug discovery the following are required of a learning machine.

Accuracy over a range of costs The cost of a missed potential drug is unknowable; the cost of predicting useless compounds as hits is the cost of subsequent screening and processing; these costs vary over time, on different projects, and according to different chemists' viewpoints.

Ability to handle unbalanced data There are typically many more inactive compounds than active compounds for any screen, thus a learning machine that always predicts a compound to be inactive will have a high overall accuracy, but will be useless in practice. Assigning a higher cost to misclassifying hits (see previous point) can alleviate this.

Fast classification The number of compounds whose activity we may desire to predict is very large, hence a short classification time is desired.

Interpretability The popularity of rule-based systems, such as decision trees, among chemists is due to their readability. If a learning machine can provide a short description of its strategy, then an expert can potentially work out what the strategy is and improve it, as well as contribute to domain knowledge.

Robustness Activity measurements are variable, and compound attributes are frequently calculated as opposed to measured. A learning machine must be robust to such variations and inaccuracies.

Ability to handle high-dimensional data Descriptions of compounds can have tens to thousands of attributes and it is often not known which are useful for prediction. If we avoid the problem of attribute selection then we require a technique that can handle many attributes, that may or may not be relevant.

Biological and chemical sense The learning machine must somehow embody biological and chemical sense, although this is difficult to evaluate, especially if the learned strategy is hard to interpret.

The following properties are obviously desirable, but not essential for the applications considered in this work.

Fast training The training set sizes are generally 10^2–10^4 (although they could potentially be $\sim 10^6$), which is not particularly large in the context of supervised learning.

Economy of space Many learning machines explicitly or implicitly perform matrix operations on $l \times l$ matrices (l is the number of training examples), if $l = 10^4$ then 1Gb of memory would be required to store such a matrix, which is likely to be impractical. For very large data sets with many examples or descriptors, it may not even be practical to store the training data in memory.

Robustness to missingness Compound properties are usually calculated and so are rarely missing. When predicting bioactivity from other bioactivities there may be missing data since not all compounds have been screened on all assays (these compounds would normally be removed from the training data, even though they do contain some information).

The support vector machine, introduced in the next section, fulfils most of these requirements to a reasonable degree, and is the learning machine of choice used in the work presented here. There are some problems when using support vector machines for pharmaceutical data analysis (and in other applications) that are discussed in Section 1.4.

1.3 The Idea of Structural Risk Minimization

When learning a classification rule in drug discovery, there are essentially two classes of compounds: those that have been screened on an assay and those that haven't. This division ignores the fact that screens are not mutually exclusive, and that compounds may have been screened against the same target at different times and under different conditions. The work presented in this thesis aims to predict activity of the unscreened compounds from the activities of the screened compounds, using some common set of descriptors. For a given task, a risk functional is specified and the aim is to minimize the risk of making predictions about unscreened compounds.

The concept of minimizing the risk incurred in using a learned decision strategy is core to supervised learning, even though it may not always be the most important consideration. An obvious approach to minimizing the true risk R is to minimize the risk on the available training data (the *empirical risk* R_{emp}). This is easily calculable since the true labels are provided for the training set (i.e. the screened compounds). Learning consists of choosing an hypothesis f from some set of hypotheses. An initial problem then is: which set of hypotheses? If we assume that all hypotheses are equally likely then we are doomed to failure since learning is impossible without some form of inductive bias (Mitchell, 1997; Popper, 1968; Vapnik, 1998). If the hypothesis space is too small then there may not be an hypothesis which accurately captures the properties of the underlying data generating process. If the hypothesis space is too large then there will be hypotheses that capture the idiosyncracies of the particular training data that are available, and thus have a low empirical risk. Such an hypothesis is unlikely to have low true risk though, since the specific idiosyncracies of the training data will be rare in the general population of unseen data. What is required is an hypothesis that captures the general properties of the training data that are repeated in the general population. Equivalently, we need to choose the right hypothesis space and then choose that hypothesis minimizing the empirical risk.

The problem of choosing the hypothesis space is termed *model order selection.* One approach to this problem is *structural risk minimization* (SRM). Statistical learning theory provides bounds on the true risk of using an hypothesis f of the form

$$R(f) \leq R_{\text{emp}}(f) + \Phi(h, l, \eta). \tag{1.6}$$

That is, the true risk is upper bounded (with probability $1-\eta$) by the empirical risk plus a penalty term dependent on the 'size' h of the hypothesis space and the number of training examples l (and on the required probability η). The 'size' of an hypothesis space is formalized in Chapter 3. The SRM approach is then:

1. Form a sequence of hypothesis spaces

$$H_1 \subset H_2 \subset \ldots$$

with increasing 'size'

$$h_1 < h_2 < \ldots$$

2. For each H_i find that f_i minimizing R_{emp}.

3. From the set $\{f_1, f_2, \ldots\}$ choose that hypothesis minimizing the upper bound on the true risk (1.6).

In practice, this is approximated by minimizing

$$R_{\text{emp}}(f) + \lambda P(f), \tag{1.7}$$

where P is a *regularization* operator and λ is a regularization constant. P is an operator chosen to capture the dominant terms in the penalty term Φ and to make the minimization tractable. (It is often the case that minimizing (1.6) directly is intractable.) The constant λ controls the trade-off between minimizing the empirical risk and ensuring that the implicit hypothesis space is not too large. The problem of determining the hypothesis space has been transformed into one of determining the optimal value of λ.

The support vector machine (SVM) implements SRM by minimizing a quantity of the form (1.7). The SVM is discussed in more detail in Chapter 3. Some of its properties are as follows.

Accuracy In terms of misclassification rate, or squared loss for regression, the SVM has been shown to be competitive with the state of the art in many applications. It can also be trained with unequal misclassificaton costs and unbalanced data.

Training time This is roughly quadratic in the number of training examples. This is not as fast as many statistical and rule-induction techniques, but is generally better than other sophisticated learning machines such as neural networks and Bayesian networks.

Classification time This is linear in the 'size' of the decision strategy. Often the decision strategy has a short description, enabling rapid labelling of objects, but in some cases the description of the decision strategy includes most of the training data, which can lead to the SVM being 'abysmally slow in classification mode' (Burges, 1998).

Interpretability The decision strategy is a weighted vote of comparisons between the new object to be classified and 'borderline' objects from the training data (known as *support vectors*). If there are few support vectors then it may be possible to gain an understanding of the decision strategy, or to construct a readable set of rules (although the meaningfulness of the support vectors is open to question).

Memory requirements The algorithms used to implement the SVM in the work presented here have memory requirements linear in the number of training examples. For these algorithms it is necessary to hold the entire training set in memory. Other algorithms are available which do not require this (although they have correspondingly longer training times).

Robustness Owing to their theoretical foundations, SVMs are highly robust to noise in the attributes and in the training labels. The SVM is not affected as much as other learning

machines by irrelevant or correlated attributes. The SVM can learn hypotheses in very high dimensional spaces without deterioration in performance.

The SVM then, seems well suited to the problems encountered in the early stages of drug discovery. There are some shortcomings however, and these are discussed in the next section.

1.4 Heuristics for Support Vector Classification

There are three main labelling problems tackled in the work presented in this thesis. Briefly, these are as follows.

HTS These data are proprietary to GlaxoSmithKline (GSK). Once a target has been identified, thousands to tens of thousands of compounds are screened against the target in a primary HTS assay. A percentage inhibition is reported for each compound. These data are coarse, one common procedure is to threshold the real-valued activities as 'hit' or 'miss'. Thresholds for these assays were provided by the screening group at GSK.

The aim of the experiments presented here is to use these labelled data to select for screening those (unscreened) compounds which are likely to be hits. This is done in two ways. The regression approach is to learn a relationship between descriptive attributes and the real-valued activity. The classification approach is to learn a rule that accurately predicts a compound to be a hit or a miss.

pK_i These data are proprietary to GSK. Once a compound has been determined active in a HTS screen, its activity is confirmed in a secondary screen. Those compounds that are active and are deemed satisfactory in pharmacological terms, are then screened over a range of concentrations to obtain an *IC50* value. These data are more refined than those from the primary HTS screen: the aim is to determine the quality of known hits. The *IC50* is a measure of concentration, thus a low *IC50* is good. Analogues of compounds having low *IC50* may also be studied further, often across related assays, e.g. to determine selectivity across a particular class of targets (such as histamines).

A pK_i value can then be determined, either by a functional assay, or calculated from the *IC50* value. A pK_i value for a compound against a particular target is essentially a constant. The effect of assay variables, principally ligand concentration, has been removed. They thus allow more direct comparison across assays done at different times, or across assays with related receptors. This latter allows for selectivity screening: it may be desired to hit one or more related receptors, or only a specific example from a class.

These data are left and right censored. To avoid throwing away useful data they were converted to a semi-quantitative ranking. Thresholds were provided for these data, for the purpose of labelling a compound as active or inactive. A classification approach was taken to predict which (unscreened) compounds would be active.

QSAR These data are available from the UCI Machine Learning Repository (Blake and Merz, 1998). This is a small focussed set of compounds, typical of those found in the traditional QSAR problems that arise in the later stages of drug discovery. The aim is to rank the compounds in terms of their inhibition of dihydrofolate reductase (reported as $\log(1/K_i)$). This is an ordinal regression problem. It is tackled by learning a ranking function great(), such that $\text{great}(n, m) = +1$ if drug n has activity higher than drug m. Thus the problem is

reformulated as one of classification. Note that this approach to ordinal regression becomes unfeasible when there are many compounds, since for l compounds there are $l(l-1)$ instances of great(n, m).

The compounds in these data sets can be characterized by a variety of attribute sets, as described in the relevant sections of this thesis. These three problems have been chosen as representative of the sort of problems typically encountered during the drug discovery process.

The objective of learning a decision rule from such data sets is to make predictions about unscreened compounds. For the purpose of validating the decision rules, some of the available data are treated as unscreened. As yet, GSK have not used the developed tools to select compounds for screening.

On the last of these problems, the standard support vector machine outperforms a range of machine learning techniques commonly used by the QSAR community.

On the second problem, that of predicting the activity of competitive antagonists, an SVM does not outperform a decision tree. Two problems are identified with the SVM.

Solution Complexity One of the oft-cited advantages of the SVM is the sparseness of the solution: the solution is an expansion on a subset of the training data, known as *support vectors* (SVs). For the competitive antagonist data, 55%–85% of the training data become SVs. This is undesirable from the point of view of model interpretability and classification speed. Further, it suggests that the SVM is overfitting the training data, and this is borne out by observing the discrepancy between training and test error. The complexity of the solution could also be interpreted as the complexity of the problem. Thus an alternative is to change the problem, by changing the training data, or the descriptive attributes.

Kernel Tuning In the absence of domain knowledge, the kernel of an SVM should be chosen to be flexible and easily computable. The Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/2\sigma^2)$ is often used. This projects the data into a countably-infinite dimensional feature space. The performance of the SVM is sensitive to the setting of σ. There are various heuristics for setting σ, or for searching for the optimal value. Although these are practical for the size of the data sets presented here, a preferable approach is to adaptively tune σ during training to reduce run-time.

These two problems are interrelated since setting σ too small will lead to a large number of support vectors and the SVM will overfit the training data.

Even when there are not many training data, it is desirable to have a fast automated method for tuning σ for the purposes of exploratory data analysis. For example, for the pK_i data, although there are only 395 compounds in the training set, there are activity data for 11 assays and at least four possible compound representations. In total there are 44 classification problems, removing one free parameter can thus save a lot of time as well as human effort. Also, if there are few SVs, there may be some interpretative value to be gained from examining the 44 sets of SVs.

A further characteristic of SVMs is that there is a global optimum. Whilst this is extremely appealing in comparison to other machine learning techniques, an observation should be made. It is often the case that an acceptable test error, or its minimum, can be achieved in far fewer iterations than it takes to attain the global optimum. Thus it is desirable to have a stopping criterion based on predicted error. Such a stopping criterion is also desirable when other heuristics are incorporated into the SVM training algorithm, since these may no longer guarantee the existence of a global optimum.

The following three heuristics are thus introduced and evaluated.

HERMES A heuristic stopping criterion based on predicted error. An estimator of the error is used to track the progress during training. Although this estimator is strictly only valid at the global optimum, its behaviour is empirically the same as the test error. Training is halted if the estimator is unchanged for a fixed number, say 100, of iterations.

LAIKA A heuristic for setting the parameter σ in a Gaussian kernel. This heuristic is applied online and subsequently refined in order to reduce training time.

STAR A heuristic for removing noisy points during training. In the standard SVM these points become SVs, i.e. they form part of the solution, this is intuitively undesirable. Removing them results in a cleaner model with fewer SVs, with little or no loss in performance.

These heuristics are described in Chapter 4 and evaluated on some publicly available data sets. In Chapter 5 these heuristics are applied to the classification of the 44 pK_i data sets.

To summarize, the application of a state-of-the-art learning machine, the SVM, to a difficult practical application has led to the development of heuristics. These heuristics trade-off various performance characteristics in order to obtain a learning machine more suited to pharmaceutical classification problems. The heuristics are general purpose and directly transferable to the application of the SVM in other domains.

1.5 Synopsis

The layout of this thesis and the main contributions (*italicized*) presented are as follows.

Drug Discovery In which the modern drug discovery process is sketched out and certain problems are highlighted, particularly focussing on the problem of representing compounds *in silico* and learning predictive functions from *in vitro* data.

The Drug Discovery Process The early stages of drug discovery consist of designing compound libraries, selecting compounds from these libraries for screening, and progressing lead compounds to the next stage. Data generated at all stages can and should be utilized to improve process efficiency.

High Throughput Screening In the early stages of drug discovery many compounds are screened in the hope of identifying active ones.

Screening Set Selection Since there are potentially trillions of drug-like compounds, an intelligent selection process is required to save time and money.

Library Design Compound libraries may be diverse, to ensure good coverage of chemical space, or focussed, for a particular pharmacological target. Intelligent filters may also be used to optimize pharmacological properties.

Representing Compounds The performance of any predictive tool learned from pharmaceutical data depends on the representation of compounds *in silico*.

Identifying Descriptive Features This is an impossible problem without domain knowledge and is not explicitly addressed in this work.

Structural Keys The presence of predefined sub-structures is indicated by a bitmap. These were developed for database searching, but have proved useful in building predictive tools.

Fingerprints Structural keys are limited in the amount of information that they enocode. Fingerprints have been developed to overcome these limitations.

Affinity Fingerprints Physicochemical and structural descriptors do not directly encode biological information. Affinity fingerprints attempt to directly encode relevant biological information.

Review of Current Approaches The traditional QSAR approach is not well-suited to the analysis of the large, noisy, heterogeneous data sets that are being generated during drug discovery. Other approaches include those based on similarity metrics. Machine learning techniques are also becoming popular.

QSAR — Rational Design For a focussed group of compounds one can construct a simple regression equation to describe the relationship between activity and physicochemical properties.

Similarity Measures The similar property principle states that structurally similar compounds will have similar physicochemical and biological properties.

Clustering Clustering aims to partition compounds into homogeneous groups, such that the groups are distinct. Compounds within a cluster may be assumed to have similar physicochemical and biological properties.

Decision Trees Compounds can be recursively partitioned into homogeneous subsets. Predictive tools can be constructed based on the labelling of these subsets.

Neural Networks Standard QSAR used linear regression, occasionally with interaction terms. Neural networks learn more complex non-linear relationships. However, they are prone to overfitting and are labour-intensive.

Pharmaceutical Data Analysis Two problems are presented: one from the very early stages of drug discovery, the other from the later stages.

Predicting the Inhibition of Dihydrofolate Reductase A small scale ordinal regression problem, typical of QSAR problems encountered in the later stages of drug discovery, is converted into a classification problem. *A manually tuned neural network exhibits the best predictive performance, but requires labour- and time-intensive model tuning.*

Analysis of HTS Data Data from the early stages of drug discovery are used to rank unscreened compounds to learn a predictive tool for selecting unscreened compounds for HTS screening. Performance is reported as the *global enhancement* of active compounds selected over random selection. *The standard linear QSAR approaches of logistic and linear regression are shown to outperform an automatically tuned neural network. The classification approach outperforms the regression approach. A 2-d representation tends to give the best results. No one combination of representation and technique is best overall.*

Support Vector Classification In which the support vector machine for classification is introduced, together with some estimators and measures of performance, and applied to a variety of representative pharmaceutical problems.

The Support Vector Machine The SVM is a powerful recent addition to the machine learning toolbox. It has shown good performance on a range of problems.

Binary Classification Supervised learning is doomed to failure without capacity control. Capacity control can be achieved by maximizing the separation of the classes (the margin), this is equivalent to limiting the 'size' of the hypothesis space.

Margin Maximization as a Quadratic Program The SVM solution is found by solving a QP with linear constraints. Hence, there is a global optimum, and parameters affecting training time do not affect the quality of the solution.

Implementation The QP is solved by a decomposition routine. Empirically, training time is quadratic in the number of training points (and linear in the number of dimensions).

Variable Misclassification Costs A simple adjustment allows the analyst to specify misclassification costs. This is important in many domains, including drug discovery.

Bounds on the Expected Error As the SVM has a strong theoretical underpinning it is possible to construct estimators of its performance without using costly validation data. *A risk functional is introduced that is a more useful indicator of performance when there are few positive examples. The $\varepsilon^{l}_{\alpha\xi}$ estimator is shown empirically to be a useful indicator of performance, although care must be taken in certain situations.*

Performance Measures The number of misclassifications made by a classifier is not necessarily a useful measure of its performance.

Receiver Operating Characteristic Curves When the costs of misclassification are variable, it is useful to plot an ROC curve to show the class-specific misclassification rates and to set the optimal operating point.

Relative Advantage The relative advantage is the proportion of positives found by the classifier, compared to random selection. This is a useful measure when the aim is to identify the small number of active compounds in a set.

Applications The SVM is applied to three problems typical of those encountered during drug discovery. Some advantages and disadvantages are highlighted.

Predicting the Inhibition of Dihydrofolate Reductase *The SVM is shown to have performance as good as the best neural network. The main advantage is the reduction in manual effort and computational time during parameter selection.*

Predicting Activity of Competitive Antagonists *The SVM does not outperform a decision tree. No improvement is obtained by using domain knowledge to select the training set, or by reweighting the misclassification costs.*

Analysis of HTS Data *A state of the art SVM does not outperform logistic regression in preferentially ranking compounds for screeening.*

Heuristics for Support Vector Classification In which the results of the previous chapter motivate the development of heuristics for: terminating training early, tuning a Gaussian kernel, and removing noisy training points.

Stopping Criteria The termination criteria for an SVM are usually defined in terms of the optimization problem. In practical data mining situations it is often desired to have stopping criteria based on time and estimated accuracy.

Persistence *The $\alpha\xi$ estimator is shown empirically to behave qualitatively as the test error during training.* This motivates a heuristic (HERMES) for early stopping of SVM training.

Results and Discussion *On some benchmark data sets, the performance of HERMES is shown to be the same as the standard SVM. The number of iterations is reduced substantially.*

Automated Model Order Selection The SVM with Gaussian kernel is very sensitive to the width σ of the kernel. A heuristic for setting σ, or tuning it, would save time and effort.

Incremental Tuning A line search on σ, or gradient descent in parameter space, leads to a good value of σ, but roughly doubles the training time.

Heuristic Approaches Heuristics from radial basis function networks suggest setting σ to be on the scale of the separation of basis function centres.

Approximating the Support Vector Set The basis function centres are the support vectors, which are not known in advance. *A number of heuristics (LAIKA) for estimating the eventual support vector set and tuning σ according to their separation are shown to find the optimal σ under the circumstances considered.*

Reducing Training Time The heuristics are modified to reduce training time.

Stopping Criterion Since LAIKA is not guaranteed to converge an alternative stopping criterion is required. *HERMES is shown to be compatible with LAIKA.*

Results and Discussion *On data sets not used in its development, LAIKA is shown to have improved performance compared to the initial heuristic on which it is based. The solutions are sparser. The training time is increased.*

Inducing Sparsity The sparseness of the SVM solution is often stated as one of its advantages. On noisy or complex data sets however, this sparsity can be lost.

Data Cleaning *A heuristic (STAR) is introduced for automatically removing noisy points during training. Noisy points become support vectors and, counter-intuitively, contribute to the decision function.*

Results and Discussion *On a number of data sets, STAR is shown to produce sparser solutions than the standard SVM with little or no loss in performance. The training time is increased, the classification time is reduced.*

Application to pK_i Data In which the heuristics introduced in the previous chapter are applied to the pK_i data sets discussed above. Data mining strategies and alternative compound representations are also assessed.

Some Data Mining Considerations Before considering modifications to the SVM, some general techniques from data mining are used to attempt to improve the performance.

Training Data Selection *Selecting the training data by clustering offers some advantages and disadvantages.*

Misclassification Costs *Altering the misclassification costs to maximize the relative advantage over randomly sampling is unsuccessful.*

Meta-Analysis *The performance with respect to two useful measures can be maximized indirectly by minimizing loss functionals. The $\alpha\xi$ estimator is not a reliable indicator of performance.*

Evaluation of Heuristics The heuristics overcome some of the deficiencies of the SVM, but sometimes at a cost elsewhere.

STAR *STAR reduces the solution complexity without loss in performance.*

LAIKA *LAIKA offers little gain on these data.*

Alternative Compound Representations The SVM is trained with three other compound representations.

Predicting from Structural Keys *Structural keys lead to less complex and more accurate solutions at the expense of longer training time.*

Predicting from Reduced Graphs *Reduced graphs offer similar advantages to the structural keys, without the increase in training time.*

Predicting from Bioactivity *A novel representation is described and shown to provide simple solutions with higher accuracy that are quicker to find.*

Finally, a discussion of the main points, critical analysis of the key contributions and results, and suggestions for future work are presented.

Chapter 2

Drug Discovery

> *Throughout history, certain plants, as well as virtually every anatomical component of animals and humans, have been ascribed some curative property: earthworms rolled in honey for the treatment of gastritis; owl brain for headache; sheep brain for insomnia; deer heart for heart disease; fox lung for tuberculosis; goat liver for jaundice; powdered human skull or the fresh blood of a dying Christian gladiator for epilepsy; rabbit testicles for bladder disease and, of course, for impotence; and cow dung for eye infections, to name but a few.*
>
> MANFRED A. HOLLINGER, INTRODUCTION TO PHARMACOLOGY

A drug is a chemical substance that that can alter or influence the responsiveness of a biological system (Hollinger, 1997). A drug either mimics, facilitates, or antagonizes a normally occurring phenomenon. According to the US Food and Drug Administration (FDA) a drug is

> a chemical which is utilized for the diagnosis, prevention, cure or amelioration of an unwanted health condition.

There are various ways of classifying drugs: physical, pharmacological, chemical, legal, and psychological depending on one's point of view. A useful categorization from a therapeutic point of view is provided by Hollinger (1997).

Drugs used to combat infection These can actually *cure* an illness, e.g. penicillin which inhibits bacterial cell wall synthesis but has no effect on mammalian cell membrane synthesis.

Drugs used to replace inadequacies of naturally occurring substances These include insulin for the treatment of *diabetes mellitus.*

Drugs that change regulation This is the largest class, concerned with treating *symptoms*, e.g. sedatives and birth control pills.

Drugs to alter mood or behaviour This class includes both licit drugs such as alcohol, caffeine and tranquilizers, and illicit ones such as cocaine and tetrahydrocannabinol (the active ingredient in marijuana).

The effect of a drug is generally caused by its interaction with a *target.* Although not all known drugs interact with a target, e.g. antacids such as bicarbonate of soda, the vast majority do. The

site of interaction may be part of a cellular constituent such as DNA, a mitochondrial enzyme, or the cell membranes.

In the context of receptor binding, the binding compound is termed a *ligand.* A compound binding to a receptor may be an agonist or an antagonist. An agonist is a ligand that binds to a receptor and promotes a physiological effect. An antagonist binds to a receptor but does not directly promote a physiological effect. Thus it may impede the action of an agonist. Note that the terms agonist and antagonist are context dependent: an agonist may be behaving antagonistically if it is reducing the binding of another agonist. The site of interaction of a drug need not be the site of binding. The drug may interact elsewhere and cause a conformational change in the receptor, impeding its function.

The effect of a compound on a target may be quantified by the strength of binding relative to a known ligand in a competitive binding assay. Alternatively, if the target is, for example, an enzyme, then the effect may be measured by the inhibition of its activity in a functional assay. The term 'percentage inhibition' is used in both cases. Ligands that are similar to each other in some sense are termed *congeners* and may be expected to have similar affinity for a given receptor (although 'similar' can typically be two or three units on a logarithmic scale).

The search for and subsequent optimization of ligands has grown from a handful per chemist per year to automated synthesis and screening of millions of compounds per year. Increasingly, statistical and artificial intelligence methods are being used to quantify and qualify the action of compounds. The need for automated modelling and decision making is now particularly acute in the early stages of drug discovery as there are too many compounds being tested for a chemist to interpret the raw data being generated.

The remainder of this chapter is organized as follows. Some historical perspectives on drug design are presented in Section 2.1, followed by a more detailed description of contemporary approaches to the early stages of drug design. In Section 2.2, various representations of compounds for building predictive models are presented and discussed. In Section 2.3, I review some statistical and artificial intelligence techniques currently used to analyse pharmaceutical data. In Section 2.4, I present some results from applying existing statistical and machine learning techniques to the analysis of pharmaceutical data. In Section 2.5, I summarize the key points and conclusions of the chapter.

2.1 The Drug Discovery Process

The approach to drug discovery through the mid-nineteenth century to the late twentieth century took as its starting point naturally occurring compounds. The initial step was the discovery of some pharmacological effect produced by a source material, occurring either in the human body or in other animals and plants. Once the active ingredient had been isolated and identified, congeners of it were synthesized and assayed. A structure-activity relationship (SAR) could then be determined. A SAR relates the biological activity of a group of related compounds to their structural features. This enables the chemist to characterize congeners of the active ingredient.

This traditional approach to drug discovery often began with the identification of a lead compound, followed by trial and error synthesis of congeners. The lead compound may be inspired by nature, knowledge of biochemical mechanisms in the body, or by the assay itself (Hollinger, 1997). The bioactive compound was first identified and this suggested a specific receptor or binding site. The pharmacological properties of the compound would then be optimized in a laborious process by synthesis and testing of variants. The traditional path of rational drug discovery can

be summarized as in Table 2.1 (Howard et al., 2001).

Modern drug discovery is taking advantage of automated, robotic *in vitro* (outside of the organism) assays to broaden the search. An advantage of *in vitro* assays is that they can be adapted to allow for screening of compounds on a mass scale, known as *high throughput screening* (HTS). HTS facilitates the screening of millions of compounds per year against many targets.

Increased screening capacity and rational drug design have led to the situation where the drug target is first identified and then a search for suitable compounds is undertaken. For example, the human genome project is now generating many potential therapeutic targets. This 'reverse pharmacology' is summarized in Table 2.2.

These and other approaches to drug discovery are not mutually exclusive. Any of them may be employed to some extent in the various stages of drug discovery and they may contribute to each other.

Predictive techniques can help to streamline the early stages of the drug discovery process by selecting compounds for screening and by identifying false positives and false negatives in HTS. Discriminatory techniques are also useful for filtering out compounds that are likely to have poor absorption, distribution, metabolism, excretion (ADME) and toxicological properties. Such techniques are, in addition, useful further downstream to characterize groups of related compounds and to identify which physico-chemical features are responsible for various desirable properties.

2.1.1 High Throughput Screening

The early stages of the drug discovery process today are concerned with identification of potential drugs (leads) and the elucidation of their mechanism of action. The lead discovery process is largely as follows (Curran, 2001). A biological target is identified by biologists working on elucidating possible disease mechanisms, e.g. an enzyme or receptor associated with a possible therapy. The first step is to identify a *natural ligand.* For example, the natural ligand for a 5-hydroxytryptophan (5-HT) receptor would be 5-HT itself (also known as serotonin). If a natural ligand is not known then a *ligand fishing* exercise is undertaken to identify one. In this case it is referred to as a *tool ligand.* It is then possible to set up a primary *in vitro* assay. Typically the compounds screened would be chosen to be as diverse as possible, in order to cover a large amount of chemical space. Compounds showing some initial activity are termed *hits.* Chemists take these initial hits and search a company's compound archives for similar molecules that could have a similar or better response. Promising molecular classes are identified and initial SARs are determined. The primary assay results may be confirmed for these compounds in subsequent screens. Substructural moieties of these molecules are then mixed and matched by the combinatorial chemists to optimize the lead compound's action.

An advantage of *in vitro* assays is that they are amenable to rapid screening of thousands of compounds. Advances in technology now enable the screening of millions of compounds a year, a process called *high throughput screening* (HTS). HTS is used to search through compound space

1. Choose a disease state.
2. Identify the active principle (natural ligand).
3. Identify the receptor or biological response.
4. Synthesize analogues to the natural ligand (drug candidates).

Table 2.1: The traditional path of rational drug discovery.

1. Choose a receptor (this may be an *orphan* receptor, i.e. its mechanism is unknown).
2. Identify the endogenous active principle, for orphan receptors this may be achieved by: testing ligands of related receptors; analysing gene expression patterns of the receptor and putative ligands; using functional assays; or, testing against families of known ligands (a HTS 'ligand fishing' exercise).
3. Elucidate the biological response mechanism and disease state.
4. Identify drug candidates for optimization.

Table 2.2: A 'reverse pharmacology' facilitated by advances in HTS and genomics.

in the hope of finding active compounds. The motivation behind this strategy is the *serendipity principle* — if we look everywhere, hopefully we'll find something. This serendipity principle is now an accepted paradigm in the pharmaceutical industry.

The goal of HTS is to find lead compounds to pass to the lead optimization phase. HTS is also used to provide an initial SAR for a lead group.

This ability to screen large numbers of compounds requires large numbers of compounds to screen. Pharmaceuticals companies have overcome the once rate-limiting step of compound synthesis by screening the vast supplies of marine and plant resources (Stead, 2000), buying compound libraries from specialist suppliers, and by combinatorial chemistry (Hijfte et al., 1999).

HTS can generate thousands of hits. A *hit* is a molecule showing activity above a given threshold in a primary screen. The threshold is a pragmatic one, it may be varied according to the assay results in order to ensure that a reasonable number of compounds are progressed. The hits from a HTS assay are subsequently screened over a range of concentrations to determine the $IC50$ value. The $IC50$ is a higher quality measurement of activity. It is the concentration of test compound required to inhibit the binding of the competing tool ligand by 50%. It is dependent on the tool ligand used, and the concentration thereof. Compounds with a low enough $IC50$ are then progressed to the next stage. (Increasingly, functional assays are being used to determine an $EC50$, the concentration of compound required to give 50% of its maximal response.)

Compounds which are isolated in chemical space are not considered progressible, since it is not possible to generate a reliable SAR. A group of related compounds that all have robust concentration-response activity is termed *progressible.* Progressible hits are modified to optimize the pharmacokinetics, ADME, toxicity and other properties. A progressible hit with sufficient potential and novelty is termed a *lead.* A lead is a sound basis for further optimization.

At this stage, the physical and pharmacodynamic characteristics of the drug are improved. Typically, solubility is increased, as is bioavailability and duration of action. The compound's ability to move through the different body fluids to the site of action is also improved, as well as increasing its potency and selectivity.

Idealized, the modern drug discovery process is designed to start from the set of 'drug-like' compounds and successively refine the search until a lead compound is found which can be optimized. There are numerous practical difficulties. The search space is vast: the lowest estimate of the number of drug-like compounds is ten trillion (Valler and Green, 2000). High-throughput synthesis and screening cannot possibly cope with this volume of compounds. When there is some prior knowledge virtual screening can be used to select compounds for screening. Virtual screening, also known as *in silico* screening, refers to using a computer program to search and evaluate very large compound libraries (Terstappen and Reggiani, 2001; Walters et al., 1998).

Even though HTS cannot possibly be used to screen everything, the volume of compounds that

are being produced by HTS is causing bottlenecks downstream. Too many potential drugs are being identified by HTS and must be filtered out at later stages owing to poor ADME and other properties. The large volume of data now being generated is also causing problems. The new field of chemoinformatics has arisen to handle the logistics of screening management and the process of extracting information and knowledge from the data (Hann and Green, 1999).

2.1.2 Screening Set Selection

In general, the development of a new drug requires the synthesis and evaluation of thousands of compounds (Eglen et al., 2000). A recurring problem in HTS is deciding which compounds to screen. This is especially true when considering new targets. Companies now have vast libraries of compounds. However, it is not economically feasible to screen all of these compounds against all targets. Compounds should be chosen for screening so as to maximize the probability of generating hits. To do this requires analysis of structure-activity relationships, either for the target of interest, if data are available, or for related targets if no screening data are available for the novel target. One simple process that is used to improve HTS is to screen a diverse set of compounds, analyse the data, and produce a decision function that indicates in what area of chemical space to look for potential hits. That is, a predictor of activity is learned for the assay and used to predict *in silico* the activity of the library compounds. Compounds predicted to have some activity can then be selected for synthesis. Building such a predictor can be treated as a classification problem: identify compounds that are more likely to be screening hits; or, as a regression problem: estimate the real-valued activity of a compound. Either approach requires that there are descriptive attributes of the compounds available that can be used in making a prediction. The goal of such an analysis would be to enrich the set of compounds chosen for screening, such that the hit rate is higher than would be obtained by random selection (Gillet et al., 1998; Dixon and Villar, 1998).

2.1.3 Library Design

As a broad definition, a chemical library is any collection of chemicals. For instance, the entire compound collection of a pharmaceutical company, a catalogue of compounds from a retailer, or a collection of hypothetical compounds existing in compound space. The last of these three is typically referred to as a *virtual library* since the compounds have not yet been synthesized. Library design is required in many stages of lead discovery. The process has been facilitated by the introduction of combinatorial chemistry (Hijfte et al., 1999), which allows for the rapid synthesis of thousands of closely related compounds, and by chemoinformatics (Hann and Green, 1999), which has arisen to handle the large number of compounds (whether synthesized or virtual) that are commonly dealt with in drug discovery.

2.1.3.1 Diverse Libraries

When initially investigating a new target there may be little prior knowledge of the mechanism of action. For example, consider a protein suggested as a target by analysis of the human genome. It may be necessary to identify compounds that bind strongly to the protein before its role in the body can be elucidated. In such cases, the ideal approach would be to test everything (Valler and Green, 2000). This is not practical, even if 'everything' is considered only to refer to a company's entire compound collection. (The entire compound collection may be screened against a particular target, but this approach is too expensive to apply it to all possible targets.) The accepted approach is to screen as diverse a set of compounds as possible (Dixon and Villar, 1998; McGregor

and Pallai, 1997). What exactly 'diverse' means is debatable, some commonly used (dis)similarity measures are given in Section 2.3.2. Once a measure of diversity has been chosen, it is necessary to maximize this criterion. This can be difficult. Clustering techniques (see Section 2.3.3) can be used to segregate the compound set into homogenous subsets that are dissimilar to each each other. Taking one member of each cluster then generates a reasonably diverse subset for screening. Dixon and Villar (1998) show that more diverse subsets can be achieved by using a stochastic search over compound space that attempts to maximize directly some measure of diversity. A diverse subset is not necessarily representative, it may consist of many outliers. An approach to selecting a representative screening set, based on clustering, is given by Menard et al. (1998).

2.1.3.2 Lead Libraries

If some knowledge is available about the target then it is desirable to incorporate this into the design of a screening library (Valler and Green, 2000). Such a library is described as *focussed* on the target of interest. Knowledge of the target may have been extracted from the results of previous (diverse or random) screening of compounds against that target. If there are known ligands for the target then these can be used to construct a three-dimensional molecular model of the part of the ligand active in binding (known as the *pharmacophore*). A search for this pharmacophore in compound databases may then be undertaken. If structural information about the target is known then docking algorithms can be used to select compounds from available databases and compound libraries (Walters et al., 1998).

An alternative approach is to take the lead compounds and generate hundreds or thousands of congeners of these. This approach is based on the principle that similar compounds have similar activity profiles. Again, the meaning of the term 'similar' may vary. Dixon and Villar (1998) give an example of screening compounds to measure binding affinity for human serum albumin. They present a set of nine compounds that are highly structurally diverse, yet have high binding affinity ($IC50$ in the range 16nM–81nM; a low $IC50$ indicates high binding affinity). For one of these compounds they select five congeners, which all have low binding affinity ($IC50 > 2.5mM$ about four to five orders of magnitude less).

2.1.3.3 Combinatorial Libraries

An ongoing process in pharmaceuticals companies is maintaining a diverse and representative compound bank. As noted above, the concepts of 'diversity' and 'representativeness' are rather fluid. If a region of chemical space is considered to be under-represented in a company's compound bank, then it may be possible to 'fill in the gaps'. If there are a handful of known compounds in the region of interest then combinatorial chemistry can be used to generate thousands of structurally similar compounds. There is no guarantee that these new compounds will be close to the original ones in property space however. It is possible to use predictive techniques to estimate the location of virtual compounds in property space (Walters et al., 1998). Compounds predicted to be in the region of interest can then be synthesized. A similar approach could be used to select for synthesis compounds with specific predicted bioactivity profiles.

2.1.3.4 Optimizing ADME properties

A lead compound may fail to become a marketable drug due to problems with absorption, distribution, metabolism and excretion (ADME) and toxicity. The absorbtion and distribution of a drug refer to its uptake and subsequent distribution throughout the body and to the site of action.

The metabolism and subsequent excretion of the drug refer to how the body processes the drug and eliminates it from the body (note that most drugs are xenobiotic and are hence removed from the body).

Filters can be developed to screen compounds for desirable ADME and toxicity properties. One early example is the 'rule of five' proposed by Lipinski et al. (1997). This states that poor absorption or permeation are more likely when:

- there are more than 5 H-bond donors;
- the molecular weight is over 500;
- $\log P > 5$, where P is the partition coefficient of the compound in an octanol-water mixture; or,
- there are more than 10 H-bond acceptors.

Certain compound classes are known to be exceptions, e.g. those that are actively absorbed into the cell. A filter for new compounds consisted of raising a 'poor absorption or permeability is possible' alert for compounds with two of the above parameters out of range. More sophisiticated techniques can be used to learn similar filters, alhough ease of interpretability is likely to be lost. Such filters can then be used to analyse compound libraries for purchase or synthesis in order to enhance the proportion of 'drug-like' compounds in a company's collection. It is important to note that there will be exceptions to any such filter. When the number of alerts is high, it is not possible for a human chemist to examine all of the suspect compounds. A filter should ideally also provide a measure of confidence in the alert, and be robust to changes in the costs of false negatives (missed opportunities) and false positives (unnecessary junk).

2.2 Representing Compounds

The way in which a compound is represented ultimately limits the success of all subsequent procedures (Kauvar et al., 1995). Dixon and Villar (1998) state the following.

> It is very difficult to conceive of structural descriptors that can reliably predict activity for a given target across the range of compounds present in a typical corporate library. If there were such descriptors, then HTS would not be nearly so widespread as it is.

In the remainder of this section, I describe the problem of selecting descriptors for the purpose of prediction. This is followed by a brief review of some descriptors that have been shown to be useful for predicting biological activity.

2.2.1 Identifying Descriptive Features

In real life applications, databases may be poorly maintained and contain irrelevant information. Not all of this can be removed by data cleaning and pre-processing. This leads to noise in the descriptors and in the target value. In pharmaceutical data analysis, the noise may vary over chemical space. For example, many descriptors are calculated by in-house or commercial software. The formulae are constructed from finite compound libraries. Such formulae may not be accurate when used to calculate descriptors of compound classes that were under-represented in the library used for constructing the formulae (see Section 2.3.1 below). There may be some compounds for which the algorithms used to compute descriptors do not provide a value. Pre-processing techniques

are thus used to clean the data, fill in missing values (if possible) and remove irrelevant features before learning a decision function from the data. Features may also be correlated or redundant, thus unnecessarily increasing the dimensionality of the input space and exacerbating the 'curse of dimensionality' (Bellman, 1961). In principle, more relevant features should increase generalization accuracy, in practice increasing the number of features may lead to a deterioration since a classifier may then model the idiosyncracies in the training data. The purpose of feature selection is to choose the optimal feature set for the learning task. The problem of feature selection may also be of direct interest itself. For example, identification of predictive physico-chemical descriptors is necessary for building predictive models and constructing SARs. When choosing a compound representation there are vast numbers of potential features. Selection of a handful of the most predictive ones can aid understanding of the prediction and modelling processes.

In the absence of domain knowledge, identifying which set of descriptors is optimal for a given task is known to be an NP-Complete problem. Ripley (1996) states 'this is an impossible problem' and is 'being supplanted by model selection methods'. That is, all of the features are used in building a model, and by controlling the expressivity of the algorithm it is hoped that irrelevant features will not contribute significantly to the model. This approach is implicit in the support vector machine (see Chapter 3).

2.2.2 Structural Keys

Structural keys were developed for high-speed searching of chemical databases (James et al., 2000). The purpose of searching chemical databases is to identify compounds with certain substructures. (Note that the term *screening* is often used in this context. The term searching will be used here to avoid confusion with *in vitro*, *in vivo* and *in silico* screening to measure or predict various biological or physico-chemical properties.) A *structural key* is a bitmap in which each bit represents the presence or absence of a specific structural feature. Examples of such structural features given by James et al. (2000) include: the presence of an element, or its frequency; important or unusual electronic configurations, such as 'triple-bonded nitrogen'; rings and ring systems; common functional groups, such as amines; and, important functional groups. A single bit may be used to indicate the presence of any of a group of rare features, to avoid excessive numbers of bits which are usually zero. Structural keys may be generic, such as MACCS keys (MACCS-II, 1994), or generated for a particular set of compounds. For example, a database of organometallics may have specific bits for metal-containing functional groups (James et al., 2000). Structural keys vary in length from a few tens to several thousands of bits. There is an obvious trade-off between specificity and length.

When learning a decision function to predict biological activity from structural keys it is important to ensure that the structural features encoded are biologically relevant. Structural keys are usually high-dimensional, sparse, binary representations. Care should be taken to ensure that an algorithm does not overfit the data or make invalid assumptions about the underlying distributions of the individual bits. Some algorithms actually seem to perform better in this kind of sparse high-dimensional space. Joachims (2001) presents theoretical reasons to support the case for using a support vector machine on such data.

Brown and Martin (1996) reported the relative ability of several structural descriptors and clustering methods to distinguish active from inactive compounds. They found that 2D descriptors gave a better separation than 3D descriptors and that the best overall was a subset of 153 descriptors from the MACCS structural key (MACCS-II, 1994). The publicly available MACCS key records the occurrence of 166 small generic and specific fragments including atom counts, ring

types and counts, augmented atoms and short linear sequences. These were chosen for substructure searching, not to reflect biological or physical properties. In subsequent work, Brown and Martin (1997) show that this ability to distinguish active from inactive compounds appears to be due to the encoding of information about the properties of a ligand relevant to receptor binding. That is, MACCS keys could be used to predict the values of properties known to be relevant to binding. The advantages of using two-dimensional structural descriptors to predict ligand binding, compared to using calculated physico-chemical properties, are two-fold. Firstly, not all of the properties may be calculable for every compound and, secondly, structural descriptors are faster to calculate (Brown and Martin, 1997).

2.2.3 Fingerprints

Structural keys suffer from a lack of generality in describing compounds. A structural key developed to represent one group of compounds, e.g. congeners of a progressible hit, may be almost useless for another group of compounds (James et al., 2000). *Fingerprints* address this deficiency by eliminating the idea of pre-defined patterns. The following description is adapted from James et al. (2000). A substructure's fingerprint characterizes that substructure, but the meaning of an individual bit is not well-defined (hence the name, in analogy with human fingerprints). Every substructure in a molecule is included in the fingerprint. Since the number of substructures is huge, it is not possible to have one bit for each substructure. Instead, each substructure serves as a seed to a pseudo-random number generator to produce a set of four or five bits. These are logically OR'd with the fingerprint of the molecule. Thus, every bit that is set in the substructure's fingerprint will be set in the molecule's fingerprint. A fingerprint doesn't indicate with certainty that a particular substructure is present, but it contains far more substructures than a structural key. Thus it is possible to do a more sophisticated search for particular desirable, or undesirable, substructures in a database of molecules.

The advantages of fingerprints over structural keys include: one fingerprinting system serves all databases and all queries; more effective use is made of the bitmap, so fingerprints are relatively dense (20%–40% of bits are on), hence a fingerprint can be much smaller than a structural key with the same discriminatory power; and, the more complex a molecule gets, the more accurately its fingerprint characterizes it. Fingerprints constructed for a particular compound collection are optimal, with about 50% of bits on. Although fingerprints were developed for substructure searching, they have been used to predict bioactivity (Brown and Martin, 1996) and physicochemical properties (Brown and Martin, 1997).

2.2.4 Affinity Fingerprinting

The descriptors described so far relate to physico-chemical or structural properties of a ligand. These contain chemical information relevant to the ligand-receptor binding (Brown and Martin, 1997) and thus may be used to predict binding affinity. However, it is often the case that compounds which are structurally similar, according to some description and similarity measure, may have quite different affinities for a given target (Dixon and Villar, 1998). To overcome this difficulty it is possible to encode directly information relevant to biological activity. Dixon and Villar (1998) give the following motivation, which should be kept in mind whenever undertaking a data mining exercise (see also Bishop (1995); Duda and Hart (1973); Fukunaga (1972); Devijer and Kittler (1982)).

> Without proper selection of descriptors, simply increasing the number of dimensions

> may tend to obscure information provided by the biologically relevant subset. ... The goal is to minimize the number of dimensions and maximize the amount of bioactively relevant information provided.

They argue that the information relevant to predicting binding is binding affinity itself. To this end they assemble a panel of about 20 functionally dissimilar proteins, so that the respective binding affinities have low correlation. An *affinity fingerprint* for a compound is the corresponding 20 affinity measurements, as determined by a high-throughput competitive binding assay (Kauvar et al., 1995). If there are no activity data for the target of interest then the affinity fingerprints can be used to select a bioactively diverse set of compounds for screening, in order to increase the chance of obtaining hits. If activity data are available then a focussed library can be selected for screening, based on correlations between the protein binding affinities and the affinity to the target.

2.3 Review of Current Approaches

The traditional approach to modelling properties of potential drugs has been quantitative structure-activity relationship (QSAR) analysis (Section 2.3.1). This aimed to generate simple equations in a few variables, in order to predict the values of physicochemical properties and biological activity. The variables were typically one-dimensional descriptors such as: number of hydrogen bond donors and acceptors, solubility, permeability, molecular weight, etc. There has also been increasing use of two- and three-dimensional descriptors, e.g. that describe the molecular surface, or the topology of a compound. Methods based on similarity measures (Section 2.3.2) and clustering (Section 2.3.3) aim to make predictions about a compound based on other compounds which are in some sense nearby in compound space. Techniques from artificial intelligence, such as decision trees (Section 2.3.4) and neural networks (Section 2.3.5), are similar to QSAR in that the aim is to learn a relationship between the target variable (e.g. biological activity) and the measured (or calculated) variables. They differ in that the relationship learned is often highly non-linear, this flexibility can often lead to improved predictive performance, but at the loss of interpretability.

2.3.1 QSAR — Rational Design

There is a widely-held assumption in pharmacology that the structural and chemical properties of a molecule are important in determining its biological activity (Bravi et al., 2000). Quantitative structure-activity relationships (QSAR) represent an attempt to predict activity from these properties. The properties of interest include binding affinities for particular receptors, ADME properties and general concepts of 'drug-likeness'. The descriptors used to characterize the compound may include one-dimensional descriptors such as molecular weight and hydrophobicity (lack of affinity for water), two-dimensional descriptors such as sub-structural fragments, and three-dimensional descriptors such as those that describe the molecular surface (considered to be key to binding of ligand to receptor). QSAR models are generally regression models in a few variables, e.g. the relationship between growth inhibition of *E. coli* by sulfonamides and sigma effect can be written as (Seydel, 1966):

$$\log\left(\frac{1}{C}\right) = 1.05\sigma - 1.28, \tag{2.1}$$

where C is the minimum concentration of the sulfonamide that inhibited the growth of *E. coli* and σ measures the relative strength of electron-withdrawing or -donating properties of a substituent.

An early example of a structure-activity relationship (SAR) was the observation by Cros, in 1863, that toxicity of alcohols to mammals increased as the water solubility of the alcohols decreased (Borman, 1990). In the 1890s, Meyer and Overton independently observed that the toxicity of organic compounds depended on their solubility in fats and oils (*lipophilicity*) (Borman, 1990; Lipnick, 1986). Louis Hammett correlated electronic properties of organic acids and weak bases with their equilibrium constants and reactivity (Hammett, 1970). The breakthrough in QSAR analysis came when Robert Muir, a botanist, attempted to correlate Hammett's electronic descriptors of a class of plant growth regulators with their biological activity. This did not lead to meaningful results until Hansch recognized the importance of lipophilicity in biological activity (Hansch, 1969).

Lipophilicity affects the bioavailability of a drug since a compound must be lipid soluble to pass through the gastro-intestinal (GI) lining. The GI tract is considered to be roughly approximated by an octanol-water partition. Hence, lipophilicity of a compound is represented by the logarithm of the partition coefficient, P, of the compound in an octanol-water mixture. Early analyses used the measured value of $\log P$ when deriving QSARs. With the increasing number of compounds available there have been numerous attempts to calculate $\log P$ from either sub-structural fragments or molecular properties (Morris and Bruneau, 2000). One of the most commonly used methods is CLOGP, based on proprietary Daylight fingerprints which encode sub-structural fragments of compounds (Leo and Weininger, 2000).

There are two factors here that may limit the predictive ability of a QSAR using CLOGP as a descriptor. Firstly, there is no reason to expect the partition coefficient in octanol-water to outperform any other solvent-water mixture in relating to processes *in vivo* (in the organism) (Morris and Bruneau, 2000). Abraham (2001) has demonstrated that for certain classes of compounds the octanol-water partition does not mimic their uptake in the GI tract, possibly owing to active uptake and other effects. Secondly, the predicted value of $\log P$ can only be expected to be accurate on compounds that were represented in the model building. Bevan et al. (2001) have patented a method for measuring $\log P$ robotically for thousands of compounds. For the 18000 compounds evaluated the measured value and the computed value had a low correlation (Spearman's $\rho \approx 0.6$). Gillet et al. (1998) give an example of data sets where accurate values of CLOGP could not be obtained.

Lipophilicity is just one descriptor commonly used in QSAR analyses. Many other calculated descriptors are also used, and these may suffer from similar problems. It is often stated that the main challenge for QSAR analysts today is developing new features and identifying useful ones (Manallack and Livingstone, 1999). The number of available descriptors has been increasing in an attempt to meet this challenge (Morris and Bruneau, 2000). The increasing number of features does not mean that new, useful (from a data miner's point of view) information has been encoded. This is particularly relevant when considering new techniques for the analysis of drug discovery data, as the process of mining data for useful information relies on their being useful information in the data to begin with.

The original aim of QSAR analysis was to model the structure-activity relationship on a small group of related compounds, e.g. congeners of a lead compound. The focus has shifted to the prediction of biological activity of new compounds. The vast majority of published work on QSAR analysis has used very small data sets of focussed compounds. These are typical of the groups of compounds of interest downstream in the drug discovery process, e.g. in lead optimization. When using such small data sets there are problems with model validation. Many authors do not provide unbiased estimates of predictive ability since the same data are used for feature selection, model

selection and building, and reporting error rates. Some recent reviews have stressed the need for proper model validation (Kövesdi et al., 1999; Eriksson and Johansson, 1996). Another aspect of using small data sets is that it is possible to measure or calculate more predictive descriptors. For example, aligned three-dimensional descriptors provide a lot of useful information but their application is quite time-consuming and they are usually restricted to datasets of less than 100 compounds (Morris and Bruneau, 2000).

In the framework of traditional drug design, QSAR analysis came to be an accepted paradigm. An active compound is identified first, and then its mediation of the receptor is investigated, together with similar compounds. The increasing demand for drugs has lead to the mass screening of compounds. QSAR analysis is still useful downstream, but must be adapted to cope with the different goals and the nature of the data now being generated in the early stages of the drug discovery process.

2.3.2 Similarity Measures

The *similar property principle* states that structurally similar molecules will exhibit similar physicochemical and biological properties (Brown and Martin, 1996). This concept can be employed to select compounds for synthesis, and subsequent screening. If no screening data are available then similarity measures can be used to identify a diverse subset of compounds for an initial screening run (see Section 2.1.2). The similar property principle can also be used for prediction of properties in the following manner. The pairwise similarity of a compound is calculated with every other compound in a dataset. The predicted value of a property for a new compound is then taken as the mean of that property for all compounds that exhibit more than a given threshold similarity to the new compound (Brown and Martin, 1997). This approach is akin to k-nearest neighbour prediction (Mitchell, 1997). To apply the similar property principle it is necessary to have a definition of similarity, or conversely of dissimilarity (e.g. distance).

Compounds are often represented as bitmaps, such as structural keys (Section 2.2.2) and fingerprints (Section 2.2.3), as described above. An obvious distance between two bitmaps A and B is the Hamming distance (Ash, 1990). This is the number of bits set to one for which two bitmaps differ, i.e.:

$$\text{Hamming}(A, B) = \#(A \text{ XOR } B). \tag{2.2}$$

where $\#(\cdot)$ counts the number of bits that are 1 in a bitmap. It is equivalent to the square of the Euclidean distance. It is normalized to remove the effect of fingerprint size to give:

$$\text{Euclidean}(A, B) = \frac{\#(A \text{ XOR } B)}{|A|}. \tag{2.3}$$

(This is not a true Euclidean distance, but this definition is used in the literature (James et al., 2000).) This distance (2.3), however, can be misleading when the bitmaps are mostly zeros. For example (James et al., 2000), consider the following two pairs of 1024 bit fingerprints: A and B have 412 bits set, of which 402 are common; C and D have 5 bits set, none of which are in common. The distance (2.3) between the first pair is the same as that between the second pair, $10/1024 = 0.0098$, yet the first pair are quite similar and the second pair are not. The distance defined by (2.3) is only a relative similarity measure.

An intuitively more reasonable similarity measure is the Tanimoto (or Jacquard) coefficient, which is given by:

$$\text{Tanimoto}(A, B) = \frac{\#(A \text{ AND } B)}{\#(A) + \#(B) - \#(A \text{ AND } B)}. \tag{2.4}$$

That is, the number of set bits in common divided by the total number of set bits (James et al., 2000). In the example above, the Tanimoto similarity of A and B is $402/412 = 0.976$, and the Tanimoto similarity of C and D is 0. The Tanimoto measure is very popular in drug design, for similarity-based prediction (Brown and Martin, 1997) and clustering (Butina, 1999).

2.3.3 Clustering

Clustering is the process of dividing a set of objects into subsets such that members of a subset are similar to each other and dissimilar to members of other subsets (Arabie et al., 1996). This requires a set of attributes to describe the objects, such that the objects can be thought of as points in some space, and a measure of similarity or distance in this space. The *centroid* of a cluster is defined to be that point which has the highest average similarity, or lowest distance, to the cluster members. For example, when clustering in a Euclidean space the centroid is the mean of the cluster members.

Clustering is used for rational screening set design (Menard et al., 1998) and property prediction (Brown and Martin, 1997). The property prediction approach is similar to that used in similarity-based prediction (see Section 2.3.2 above). A test compound is clustered with compounds for which the property is known. The clustering is performed with respect to descriptors that are known for all compounds, e.g. structural keys. The value of the property of the test compound is predicted to be the mean value over other cluster members (Brown and Martin, 1997). The underlying assumption is that the similar property principle holds.

Clustering is used in rational screening set design in two ways. If no activity data are available then clustering is used to select a representative subset of compounds for screening (McGregor and Pallai, 1997; Menard et al., 1998). The idea is that to maximize the number of hits (or, indeed, to obtain any hits at all) on an assay for a target with unknown structure the compound space should be sampled diversely. The advantages and disadvantages of this approach are discussed in Valler and Green (2000). If activity data are available for a target then clustering can be used to predict activity in a manner analogous to the prediction of properties. A set of unscreened compounds is clustered with a set of screened compounds. Those compounds that are clustered with compounds previously found to be active are then considered for screening. The clustering should hopefully separate the actives from the inactives and cluster the actives together (Brown and Martin, 1996). A drawback to this approach is that there is little chance of breaking out and finding something novel — a New Chemical Entity — that is patentable.

One of the most popular clustering methods in drug discovery is due to Jarvis and Patrick (1973). A similarity, or distance, measure is specified. Two parameters, j and k, control the clustering. The j nearest neighbours of each compound are calculated. Two compounds are then clustered together if (and only if) they are in each others j nearest neighbour list and have k of those nearest neighbours in common. Brown and Martin (1996, 1997) show that this technique performs poorly in comparison to others, including an agglomerative method (Ward, 1963) and a divisive method (Guénoche et al., 1991), in both predicting properties and screening set selection. Jarvis-Patrick clustering is very sensitive to the values of j and k (Menard et al., 1998) and tends to produce a few large clusters and many singletons. Singletons are undesirable since they are apparently not representative of anything else. Subsequent developments have been made in attempts to remedy the problems of Jarvis-Patrick clustering (Menard et al., 1998; Butina, 1999). Jarvis-Patrick clustering continues to be popular, despite its deficiencies, as it is very fast.

2.3.4 Decision Trees

Decision tree learning, also known as recursive partitioning, can provide an informative model, through which predictive rules are induced to solve classification problems (Breiman et al., 1984). The method uses a process called recursive partitioning. In their simplest form, e.g. C4.5 (Quinlan, 1992), each attribute of the data is examined in turn and ranked according to its ability to partition the remaining data. The data are propagated along the branches of the tree until sufficient attributes have been chosen to correctly classify them. The trained classifier has a tree-like structure. Each 'leaf' of the tree represents a subset of the data that lies wholly in one class. Decision trees have a tendency to overfit the training data. In the presence of noise, the tree can be pruned to allow for misclassifications in the training data. Pruning is a heuristic process, usually requiring the use of a hold-out validation set to determine the optimal tree structure (Breiman et al., 1984). There may exist several tree structures capable of achieving a given training accuracy. Decision tree learning algorithms are biased to select the simplest of these trees. Decision trees can also be constructed where the branching criteria are allowed to be linear or non-linear combinations of attributes, instead of just splitting on a single attribute. These methods, e.g. OC1 (Murthy et al., 1994), are more powerful but require extensive model selection. These methods produce shorter trees, with more complex branching rules.

Decision trees, and other algorithms that compute rule sets, are popular with medicinal chemists owing to their interpretability (Hann and Green, 1999). This interpretability can be enhanced by a careful choice of features. It is possible to generate decision trees and rule sets from the solutions provided by more powerful techniques such as neural networks and support vector machines (Fellenz, 2001).

2.3.5 Neural Networks

Neural networks are a biologically inspired form of distributed computation (Bishop, 1995; Ripley, 1996; Hertz et al., 1991). In a neural network, there are a large number of interconnected nodes that perform summation and thresholding, in loose analogy with the neurons of the brain. Feed-forward neural networks can be used to learn a real-valued or a discrete-valued decision rule. Neural networks do not explicitly model the underlying distribution but in certain cases the outputs can be treated in a rigorous probabilistic manner (Bishop, 1995).

Neural networks have been successfully applied in many settings, including speech recognition (Lang et al., 1990), handwritten digit recognition (LeCun et al., 1989) and driving a car (Pomerleau, 1993). Nevertheless, they suffer from many problems and are not well controlled learning machines (Vapnik, 1998). Neural networks have been proposed as a useful tool for many areas of drug design (Schneider, 2000), and have exhibited good performance in optimizing chemical libraries (Sadowski, 2000) and SAR analysis (Kövesdi et al., 1999). However, in many cases the reasoning behind choosing a particular architecture or training algorithm is unclear, and model selection and training times are rarely reported. This is not too important in small-scale drug design problems, but will become increasingly significant with the growth of pharmaceutical data sets. Below I describe the perceptron and some extensions, the multi-layer perceptron and the radial basis function network.

2.3.5.1 Multi-Layer Perceptrons

The perceptron (Rosenblatt, 1958) is an information processing unit of the form:

$$f_j(\mathbf{x}, \mathbf{w}) = \text{sgn}\left(\sum_i w_{ji} x_i - \theta_j\right). \tag{2.5}$$

where $\text{sgn}(z) = +1$ if $z > 0$ and $\text{sgn}(z) = -1$ if $z < 0$[1]. The output of a perceptron at node j, f_j, is the weighted sum of its inputs, x_i, thresholded at θ_j. Such a perceptron is able to represent all decision hyperplanes over the space of its inputs $\mathbf{x} \in X$. In order to learn more complex decision functions the inputs x_i are fed into a number of perceptrons nodes, each with its own set of weights and threshold. The outputs of these nodes are then input into another layer of nodes and so on, until the output of the final layer of nodes is the output of the network. Such a network is termed a *multi layer perceptron* (MLP) and the layers of nodes whose input and output are seen only by other nodes are termed *hidden*. An MLP with enough hidden layers and nodes can approximate any decision function to an arbitrary degree of accuracy (Bishop, 1995), but this result is not much use in practice as it does not specify the architecture of the network or the values of the weights and thresholds.

Normally the architecture of an MLP is specified in advance and the weights and biases are estimated by supervised learning. In order to propagate errors through the network the sgn function in (2.5) is replaced by a differentiable sigmoid transfer function such as tanh. See Hertz et al. (1991); Bishop (1995) for a discussion of supervised training algorithms for MLPs.

MLPs suffer from many problems and are not well controlled learning machines (Vapnik, 1998). Training can take many iterations to converge and the speed is dependent on a number of parameters which specify the learning rate, a momentum term and stopping criterion. Also, MLPs are prone to over-fitting without some sort of capacity control (see Chapter 1, Section 1.3). Capacity control methods for NNs suffer from requiring a number of parameters to be set that affect the rate of convergence and the quality of the solution. Model combination (Drucker et al., 1993; Breiman, 1994) and Bayesian methods (Neal, 1996) can partially overcome these deficiencies, but require many models to be trained and are hence computationally expensive. Even so, as noted at the beginning of this subsection, neural networks have shown good performance in practice.

2.3.5.2 Radial Basis Function Networks

A Gaussian radial basis function classifier is of the form:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} w_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2\sigma_i^2}\right) + b\right), \tag{2.6}$$

where b, σ_i are constants. That is, a new point $\mathbf{x}$ is compared to a collection of prototypes $\mathbf{c}_i, i = 1, \ldots, m$, the similarity measures, in this case Gaussian, are weighted and summed. If this quantity is not less than $-b$ then $\mathbf{x}$ is assigned to class 'true', otherwise to class 'false'. Such a weighted sum of Gaussian bumps can approximate any decision boundary arbitrarily well, given enough basis centres $\mathbf{c}_i$ and appropriate values for the weights w_i and the bias b. Given a collection of centres the weights and bias can be computed by gradient descent as for an MLP (Bishop, 1995). The RBF centres $\mathbf{c}_i$ and widths σ_i are usually chosen in advance. Typically this is done by k-means clustering. Training an RBF network is a two-stage process, in contrast to training an MLP, for

[1]The case $z = 0$ can be assigned arbitrarily as ± 1, randomly, or left as 'don't know'.

which all weights are optimised simultaneously. The quality of an RBF solution thus depends on the quality of the clustering, which depends on the clustering algorithm and the choice of k.

2.4 Pharmaceutical Data Analysis

In this section, I describe the application of some of the techniques outlined above on two problems characteristic of those found in drug development. The first problem consists of predicting the inhibition of dihydrofolate reductase by pyrimidines (this was the only publicly available QSAR data set at the time of study). Four algorithms were used to learn neural networks of differing architectures and functionality. These are compared to a decision tree. The neural networks required extensive model tuning and training times in comparison to the decision tree, but, in general, gave better performance. The second problem is to rank compounds in order such that the compounds screened first are more likely to be active. If such a ranking were available then fewer compounds would need to be screened to obtain a given number of hits. Alternatively, if there are resource limitations, screening the n compounds that are most likely to be hits will give an enhancement over selecting compounds randomly. This problem can be treated as a classification problem or as a regression problem. These two approaches are evaluated using both standard statistical techniques and neural networks.

2.4.1 Predicting the Inhibition of Dihydrofolate Reductase

The data used in this experiment were obtained from the UCI Data Repository (Blake and Merz, 1998) and are described in King et al. (1992). The problem is to predict the inhibition of dihydrofolate reductase by pyrimidines. The biological activity is measured as $\log(1/K_i)$, where K_i is the equilibrium constant for the association of the drug to dihydrofolate reductase. Such QSAR problems are generally formulated as regression problems, i.e. learn the conditional density of the target, y, given some predictive attributes, $\mathbf{x}$. In this case the target is $\log(1/K_i)$ and the attributes are as follows. Each drug has three positions of possible substitution. For each substitution position there are nine descriptors: polarity, size, flexibility, hydrogen-bond donor, hydrogen-bond acceptor, π donor, π acceptor, polarizability and σ effect. Each of the twenty-four non-hydrogen substituents in the compound set was given an integer value for each of these properties (see King et al. (1992) for details); lack of a substitution is indicated by nine -1s. This gives twenty-seven integer attributes for each drug. This approach is adapted to avoid solving a regression problem by recasting it as one suitable for classification or inductive logic programming (ILP), as described in King et al. (1992). Here we focus on the classification problem.

The task is to learn the relationship great(d_n, d_m) which states that drug d_n has a higher activity than drug d_m.[2] Each instance consists of two drugs, giving fifty-four attributes in total, and a label 'true' or 'false', indicating the value of the relationship great(). There are 55 compounds in the data set. In order to obtain a good estimate of the performance of the algorithms, the data were partitioned into a five-fold cross-validation series as follows. The 55 compounds were randomly partitioned into 5 sets of 11 compounds. Each fold consisted of a training set A of 44 compounds and a test set B of 11 compounds. The classification training data were constructed by labelling each pair $(n, m) \in A \times A$ according to the true value of great(d_n, d_m). This gives $44 \times 43 = 1892$ examples of great(). In practice, however, the training sets were slightly smaller since great() is not defined when the two drugs have the same activity. The classification test data

[2]The natural transitivity of the relationship great() is not exploited when learning nor when making the predictions.

were constructed by forming all pairs $(n, m) \in (A \times B) \cup (B \times A) \cup (B \times B)$ (i.e. all comparisions not in the classification training set) which gives slightly less than $44 \times 11 + 11 \times 44 + 11 \times 10 = 1078$ examples of great().

Owing to the above construction each instance in the classification data was followed by its inverse, for example if the first instance represents great(d_2, d_1) = true then the second instance represents great(d_1, d_2) = false. This produced a symmetric two-class classification problem where the classes were equal in size and had equal misclassification costs.

Each of the training sets was then used to train an algorithm to produce a classification rule. This classification rule was then used to make predictions on the test set. There are two types of test comparison: between a test compound and a compound in the training set, and between two test compounds. There are 550 comparisons of the latter type. For each pair (n, m), the former type occurs twice during the five-fold cross-validation: once when $n \in A$ and $m \in B$, and once when $m \in A$ and $n \in B$. There are 4840 such test comparisons. Thus there are 5390 test comparisons in total. The future probability of mislabelling a pair may depend on whether or not one of the pair was in the training set. If we intend to use the rule to rank a large number of new compounds, then there will be many comparisons between new compounds. If we intend to incorporate only a few compounds into the ranking learned for the training compounds, then most comparisons will be with only one or two training compounds. In the results presented here, I assume the two misclassification rates are equal. A further point is that the test comparisons are not independent within each fold or across folds. Mislabelling (n, m_1) is not independent of mislabelling (n, m_2). Nor is mislabelling (n, m) when $n \in A$ and $m \in B$ independent of the converse. One approach would be to estimate all of the probabilities involved. However, with only 55 compounds, this is infeasible. The approach taken here is to assume that mislabelling (n, m_1) is independent of mislabelling (n, m_2) and the mislabelling (n, m) is the same whether or not n or m is in the training set. I also assume that mislabelling (m, n) is equivalent to mislabelling (n, m). The cross-validated error rate, ε, is thus reported as the proportion of the 5390 test pairs which are mislabelled and the standard error is calculated as $\sqrt{\varepsilon(1-\varepsilon)/1485}$, where 1485 is the number of essentially distinct comparisons under these assumptions.

Finally, note that in learning the relationship great() the compounds could be ranked in terms of their activity without going to the effort of predicting that activity. This is in contrast to the more general and harder problem of learning a set of rules or a regression equation to predict the real-valued activity.[3]

2.4.1.1 Algorithms

Full details of the implementation and the various parameter settings are given in Burbidge et al. (2000). All of the algorithms were implemented in the data mining package CLEMENTINE 5.1 (SPSS, 1999). Below I summarize the salient details.

The C5.0 algorithm of Quinlan's was used to learn a ruleset to classify the data. This algorithm is a modification of the well-known C4.5 algorithm (Quinlan, 1992). This algorithm required no model selection (which is to say, after a bit of experimentation it appeared that the defaults were optimal).

Four neural networks were also trained, as described below.

MLP A two-layer sigmoid network was trained with the number of nodes in the hidden layer varied from 2 to 54, in steps of 2. A validation set of size 20% was held out from the training

[3]The regression approach has recently been shown by others to be successful. This work has not yet been published.

Algorithm	Error	s.e.	Time/s
MLP	0.1381	0.0090	857
'Dynamic'	0.1488	0.0092	613
'Prune'	0.1620	0.0096	692
C5.0	0.1870	0.0101	4
RBF	0.2272	0.0109	199

Table 2.3: Predicting the inhibition of dihydrofolate reductase: Estimated generalization errors and training times averaged over the five cross-validation folds. The manually tuned MLP has the best performance, but the longest training times. The RBF performed badly. (Reproduced from Burbidge et al. (2000).)

Algorithm	Time/s
MLP	20715
RBF	11719
'Prune'	13338
'Dynamic'	4748

Table 2.4: Predicting the inhibition of dihydrofolate reductase: Model selection time for the neural networks. No model selection was performed for C5.0. The manually tuned neural network required the most computation time, and this led to the best performance (see Table 2.3). (Reproduced from Burbidge et al. (2000).)

data to estimate the optimal number of hidden nodes (viz. 20).

Dynamic Starting from a minimal network, nodes were dynamically added during training. A validation set of size 20% was used to estimate the optimal architecture.

Prune Starting a from a large network, nodes were selectively pruned if they did not appear to contribute significantly to the performance of the network (again, on a validation set). This method includes the possibility of removing input nodes, i.e. it can perform feature selection. However, the set of features removed varied over folds, suggesting that this method is unstable.

RBF k-means clustering was used to select the RBF centres and widths. An RBF network was then trained to classify the data. The number of clusters (i.e. hidden nodes) was varied in steps of 2 from 2 to 54. The optimal number of clusters (viz. 14) was chosen as that minimizing the error rate on a hold-out validation set.

Note that for the neural networks, once the optimal architecture had been estimated, the networks were retrained using all of the available training data.

2.4.1.2 Results and Discussion (QSAR)

The estimated generalization errors and training times reported by Burbidge et al. (2000), averaged over the five cross-validation folds, are reproduced in Table 2.3. Also shown are the standard errors; in the following, the one-tailed t-test at 95% confidence is used. If speed were the key issue then the C5.0 decision tree would be the preferable algorithm. However, this had significantly worse generalization accuracy than the manually tuned (MLP) and 'Dynamic' sigmoid neural networks. The RBF network was significantly worse than the other algorithms on these data, which may be due to the quality of the clustering. The MLP was significantly better than the other algorithms, except for 'Dynamic'.

Four different neural network architectures were used on these data, all of them requiring heuristic methods for model selection which are computationally intensive. The total times for model selection and validation are shown in Table 2.4. Note that once a neural network architecture has been selected on the basis of the hold-out set it was then retrained using all of the training data. Again, if speed were the most important issue then C5.0 would be chosen as model selection provided no increase in accuracy in this case. This not a general rule, however, as decision trees usually require extensive model selection to avoid overfitting (Breiman et al., 1984). I will return to this problem later in Chapter 3 (Section 3.3.1), where the SVM is used to classify these data.

2.4.2 Analysis of HTS Data

The screening of large numbers of compounds produces a large amount of data. Typically, there may be thousands to tens of thousands of compounds for which activity data are generated. In competitive binding assays, results are usually given as a percentage of the relative inhibition of the tool ligand. In HTS the compounds are present in minute quantities which can lead to processing problems (e.g. liquid handling) and results can be variable. Higher throughput of screens leads to a reduction in the precision of this activity measurement. The measured inhibition is usually thresholded at some pre-specified value in order to label the compounds as a hit or miss. Many of the 'hits' from HTS turn out to be inactive. If the threshold were increased then there would be fewer of these false positives, but there would be an increased risk of missing potential opportunities. The threshold is thus a pragmatic one.

The standard QSAR approach is to learn a regression function and hence these techniques are not immediately applicable to analysis of higher throughput data. The problem lies not with the concept of QSAR itself, but with the statistical techniques used in the analysis. A further complication with these data is the significant error rate. Compounds that are predicted active by an HTS screen may turn out to be inactive upon further testing (false positives), more worryingly, HTS may miss active compounds (false negatives). There have been some attempts to characterize the reliability of HTS (Zhang et al., 1999, 2000). Traditional QSAR techniques are very sensitive to outliers and errors and this consideration must be taken into account when constructing a predictive tool based on these data (Labute, 1999).

Note that the QSAR techniques of linear and neural net regression are not usually used in the analysis of HTS data in the way presented below. They were included in the following analysis to serve as a benchmark for the techniques introduced and developed in the thesis.

2.4.2.1 Measuring Performance

Below, the classification and regression approaches to analysing HTS data are considered. In order to compare the two approaches, and to provide a more useful measure of performance with respect to the compound selection process, the following performance measure due to Gillet et al. (1998) is used. The $A50$ is defined to be the number of compounds that must be tested to find half of the active molecules. Gillet et al. (1998) define the *global enhancement* (GE) as the ratio of the $A50$ expected for random selection (which would be half of the compound set) to the actual $A50$. This is a more practical measure of performance than reporting error rates or risk for either classification or regression. It also provides a meaningful comparison between the classification and regression approaches. To calculate an $A50$ requires a ranking of compounds according to activity (cf. Section 2.4.1) — compounds can then be screened in order of predicted activity. The regression approach provides this ranking immediately. The classification approach provides a labelling with

Assay	No. Compounds	π^s_{+1}
a1	8835	0.0862
a2	3533	0.0436
a3	2151	0.0604
a5	8733	0.0641
a6	3875	0.0456
a7	7990	0.0354
a8	7467	0.0371

Table 2.5: HTS Statistics. The number of compounds screened on each assay (for which features were available) is shown with the hit rate π^s_{+1} (the proportion of positive examples in the training sample) for each assay.

a confidence value that can be used to rank the compounds. CLEMENTINE's logistic regression provides a class probability and the neural network classifiers provide a confidence measure in their predictions.

2.4.2.2 Results and Discussion (HTS)

The data used for this analysis were generated by high-throughput, competitive binding assays. The targets were all G-protein coupled receptors (Howard et al., 2001), which form the largest and most diverse group of transmembrane proteins involved in signal transduction. That is, they are all proteins that span cell membranes in humans and are involved in the transport (transduction) of various chemicals into and out the cell. The screening is performed robotically by pipetting the receptor into a well with a tool ligand, which binds to the receptor (i.e. it behaves agonistically). The compound under test is then introduced and the reduction in specific binding (i.e. inhibition) of the agonist is determined. For example, the tool ligand may be radio-labelled, in which case the amount of free radioligand corresponds to the level of inhibition.

Activity data for seven HTS assays for a total of around 9000 compounds were available. Not all compounds have been screened in all assays. The number of compounds screened, for which features were available, in each of the assays is given in Table 2.5, together with the hit rate for each assay. The hit rate is the proportion of compounds having activity greater than a threshold set by the screening group. The data sets are all unbalanced, with the ratio of inactives to actives varying from 11:1 to 27:1. Hence, predicting all compounds to be inactive achieves between 91% and 96% accuracy rate. Classification accuracy is not a useful measure of performance. The global enhancement *GE* defined above (Section 2.4.2.1) is more suitable.

For the purpose of obtaining unbiased estimates of the true performance, the data sets are partitioned into training and test sets, in the ratio 3:1. The training set is further partitioned 1:1 to provide a validation set during model building.

There were three feature sets, labelled fs01, fs02, and fs03, provided by SmithKline Beecham (now GlaxoSmithKline). The first feature set, fs01, consisted of a 55 variable vector representing variable-length molecular structure information that had been tranformed to a fixed length using M. Waegner's autocorrelation methods. The second feature set, fs02, consisted of one attribute, CLOGP (Leo and Weininger, 2000). The third feature set, fs03, consisted of 12 attributes computed from partial atomic charge information for a single three-dimensional conformation of the molecule. These were the polar moments of inertia scaled by the number of atoms, the ratio of molecular volume to the molecule's bounding box, and the dipole components projected onto the principal axes.

The goal of the analysis is to rank unscreened compounds in order of activity. The compounds can be then screened in order. The *GE* measures the increase in hits detected over random selection of compounds for screening. This can be treated either as a regression problem or as a classification problem. The regression framework is the most natural since the activities are real-valued. However, owing to the uncertainty in the measurements when screening large numbers of compounds the problem is usually converted to one of classification by thresholding the activity at some level specified by the screening group.

Four algorithms, implemented in CLEMENTINE 6.0 (Clementine 6.0, 2001), were used to rank the compounds in order of activity. The algorithms were as follows.

Linear Regression This learns a linear relationship between the features and the activity, under the assumption that the noise in the activity measurement is independent normal (Smith and Draper, 1998). Regression models are fast to train and well understood.

Logistic Regression This learns a relationship between the input fields and the probability of being active (Ripley, 1996). This can be used to rank compounds according to the probability that they are hits, without actually estimating the real-valued activity.

Neural Network Regression (NNR) This learns a non-linear equation in the features that predicts the activity (Bishop, 1995). Dynamic neural networks were used, i.e. the architecture is optimized automatically by using 20% of the training data for validation.

Neural Network Classification (NNC) This learns a non-linear discriminant to identify active compounds (Bishop, 1995). The distance from a data point to the separating surface is used to give a measure of the confidence in a prediction. This can then be used to rank the compounds in order of predicted activity. Dynamic neural networks, optimized automatically as above, were used.

CLEMENTINE defaults were used for linear regression and logistic regression. The dynamic neural networks were stopped when there had been no change in predicted performance for 200 cycles through the training data.

The global enhancement over random screening of the four techniques is shown in Tables 2.6–2.8. The linear techniques outperformed the neural networks in 20 out of the 21 tasks. Logistic regression was better than linear regression on five out of seven of the tasks using feature set fs01 (Table 2.6), and on four out of seven using fs03 (Table 2.8). On feature set fs02 (Table 2.7) the performance of the two linear techniques is equal except for assay a6 (and for assay a2, just), where logistic regression wins. The best *GE* achieved for each assay is shown in Table 2.9, with the corresponding feature set and algorithm. Logistic regression wins five out of seven times (but is tied on a8). Feature set fs01 wins four out seven times. The neural networks do badly on these data, which implies that more sophisticated model selection is required when using these more flexible techniques.

Note that the standard QSAR approach of linear regression would not normally be used for these data. The purpose of the above analysis is to provide a reference benchmark against which to compare other algorithms.

2.5 Summary

The work presented in this chapter concerns the application of supervised learning techniques to facilitate the early stages of the drug discovery process.

Assay	Linear	Logistic	NNR	NNC
a1	1.18	**1.51**	1.14	1.12
a2	1.87	**2.48**	0.93	1.19
a3	**2.35**	1.92	1.20	0.81
a5	1.79	**2.17**	1.01	0.96
a6	3.00	**5.21**	0.81	1.39
a7	**2.15**	1.72	0.79	1.02
a8	2.82	**4.15**	1.00	1.22

Table 2.6: HTS Results: fs01. The highest global enhancement for each assay is emphasized. Logistic regression wins five out seven times. Across these assays, the neural networks are not better than random selection on average and never win in comparison to the linear techniques.

Assay	Linear	Logistic	NNR	NNC
a1	**1.27**	**1.27**	0.86	0.98
a2	2.56	**2.57**	0.93	0.91
a3	**3.27**	**3.27**	1.54	1.54
a5	**1.37**	**1.37**	0.97	0.97
a6	0.84	**1.28**	1.22	1.27
a7	1.08	1.08	**1.18**	**1.18**
a8	**4.54**	**4.54**	0.93	0.96

Table 2.7: HTS Results: fs02. The highest global enhancement for each assay is emphasized. The linear techniques win six times out of seven. For the linear techniques, classification and regression have approximately equal performance. This also holds to a slightly lesser extent for the neural networks. Across these assays, the neural networks are not better than random selection on average.

Assay	Linear	Logistic	NNR	NNC
a1	1.09	**1.26**	1.25	0.92
a2	**2.09**	2.07	0.92	0.92
a3	**4.37**	3.49	0.81	0.62
a5	1.22	**1.35**	0.99	0.97
a6	1.76	**2.93**	1.05	1.16
a7	1.64	**1.66**	1.36	0.90
a8	**4.32**	3.65	4.04	0.77

Table 2.8: HTS Results: fs03. The highest global enhancement for each assay is emphasized. Logistic regression wins four out of seven times. Across these assays, the neural networks are not better than random selection on average and again never win.

Assay	*GE*	Algorithm	Features
a1	1.51	Logistic	fs01
a2	2.57	Logistic	fs02
a3	4.37	Linear	fs03
a5	2.17	Logistic	fs01
a6	5.21	Logistic	fs01
a7	2.15	Linear	fs01
a8	4.54	Log./Lin.	fs02

Table 2.9: HTS Results: Summary. The highest global enhancement (GE) for each assay is shown with the corresponding algorithm and feature set. The linear techniques win on all of the assays. Logistic regression wins four out of seven times and is tied on a8.

- The biological activity of a compound is related to its physicochemical properties and structural features.
- Advances in robotics allow mass screening of compounds against targets. Idealized, the early stages in the drug discovery process today are as follows.

  1. A target is identified.
  2. Library design methods are used to select compounds (i) for synthesis from virtual libraries, (ii) from in-house collections, (iii) from external suppliers.
  3. Tens of thousands of compounds are screened against the target at a single concentration.
  4. 'Hits' from primary screens are identified and screened over a range of concentrations to determine an $IC50$, a measure of binding affinity.
  5. Low throughput assays are used to determine the functional activity of the confirmed hits. Progressible hits, i.e. those representative of a compound series with acceptable activity (i.e. binding affinity and functional activity) are identified and screened for ADME and toxicity properties.
  6. Non-toxic, novel, progressible hits with good ADME properties become leads, i.e. they have sufficient potential to progress to a full drug development program.

There are too many drug-like compounds to synthesize and screen them all, even with projected advances in robotics and miniturization. Data analysis can be used to select compounds for screening, and to characterize compounds and targets. The aim of the experiments reported in this thesis is to predict biological activity from screening data. This could improve efficiency in the early stages of drug discovery.

The contributions to supervised learning presented in this thesis have been motivated by the problems encountered when analysing the data generated during the early stages of drug discovery. The key points of the data analysis presented in this chapter are as follows.

- QSAR analysis attempts to model biological activity as a function of physicochemical descriptors for a small group of related compounds.
- HTS data have characteristics that prevent the use of standard QSAR techniques.
- An evaluation of linear and non-linear predictive techniques on high-throughput screening data established the following.

  - Standard statistical techniques outperformed automated neural networks on these data. However, these traditional QSAR techniques are still considered inadequate. This suggests that more sophisticated model selection is required for the neural networks.
  - Logistic regression slightly outperformed linear regression, i.e. it was easier to rank the compounds in terms of activity than to predict activity.
  - There appeared to be no relationship between the hit rate of an assay and the performance of the predictive techniques.
  - The enhancement over random selection varied from 1.5 to 5.2. This can be compared to results of (Gillet et al., 1998) where an enhancement of 3.1 was achieved for a comparable data set (although the target was 'drug-likeness' and not any specific activity).

- Some standard intelligent data analysis techniques were evaluated on a small QSAR problem that had been recast as a ranking problem. The following conclusions were drawn.
  - The C5.0 decision tree was fast to train but gave poor performance.
  - Radial basis function networks had surprisingly poor predictive performance.
  - Manually tuned sigmoid neural networks outperformed the other techniques, including automated neural networks, but required an extensive model building phase.

The analysis of data generated during the early stages of drug discovery poses many challenges. On one problem, non-linear machine learning techniques performed well, but required a lot of human intervention. On another, more difficult, problem, standard linear techniques performed better than the more powerful techniques. Better learning machines are desired in the pharmaceutical industry for both of these, and other, problems.

Chapter 3

Support Vector Classification

Neural networks are the second best way of doing just about anything.

JOHN DENKER, NEURAL NETWORKS FOR COMPUTING

The support vector machine (SVM) was introduced by Vapnik and co-workers (Vapnik, 1995; Cortes and Vapnik, 1995) initially as a learning algorithm for binary classification by hyperplanes in the input space. The motivation behind the SVM is that of *structural risk minimization* based on statistical learning theory (Vapnik, 1998). This provides a principled way to quantify and minimize the expected risk of using a classifier on unseen data. For linearly separable classes, the basic idea of the SVM is to maximize the distance between the decision hyperplane and the nearest data points, this distance is termed the *margin*. Intuitively, if the hyperplane is constructed to be in the middle of the separating band then the SVM is likely to perform well. Statistical learning theory quantifies this concept of the margin. Restricting the space of hypotheses to those which separate the data with a large margin reduces the size of the hypothesis space and hence prevents over-fitting of the available data. This basic idea has been extended to allow for training errors in classification.

The solution to the SVM is found by solving a convex quadratic program (QP) which has a global optimum. The learning parameters of the QP solver affect only the time and space complexity and have almost no effect on the quality of the SVM solution. This leaves very few free parameters. These can be optimized by applying the risk bounds of statistical learning theory without the need to reserve a portion of the data for model validation.

The SVM has been generalized to allow non-linear decision surfaces by learning an hyperplane in some *feature space* that is the image of a non-linear mapping of the input space (Aizerman et al., 1964). This is achieved by constructing a *kernel* function that acts as the interface between the data and the learning algorithm. The SVM has been shown to be competitive with existing algorithms on a range of classification tasks including handwritten character and digit recognition (Cortes and Vapnik, 1995), text categorization (Joachims, 1998) and gene classification (Brown et al., 2000).

In the remainder of this chapter, I describe the support vector machine for classification, its implementation and some estimates of its performance (Section 3.1). In Section 3.2, I describe the receiver operating characteristic curve and relative advantage as measures of performance. In Section 3.3, I present some results for support vector classification of pharmaceutical data. Finally, in Section 3.4, I summarize the salient points and conclusions of the chapter.

3.1 The Support Vector Machine

In this section, I describe the problem of binary classification by hyperplanes and define the optimal separating hyperplane, which is the SVM solution (Section 3.1.1). I then briefly outline the formulation of the SVM as a quadratic program (Section 3.1.2) and discuss some implementation issues (Section 3.1.3). More details can be found in the tutorials by Burges (1998) and Osuna et al. (1997b) and in the book by Cristianini and Shawe-Taylor (2000) (see also Burbidge and Buxton (2001)). In Section 3.1.4, I briefly outline the extension to the case of unequal misclassification costs which are frequently encountered in real life classification problems such those occurring in chemoinformatics. In Section 3.1.5, I describe some estimates of the expected generalization error of an SVM, together with a brief analysis of their behaviour on real-world data sets.

3.1.1 Binary Classification

Consider learning a binary decision function on the basis of a set, S, of training data drawn independently and identically distributed (i.i.d.) from some unknown distribution $p(\mathbf{x}, y)$,

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\},\ \mathbf{x}_i \in \Re^d,\ y_i \in \{-1, +1\}. \tag{3.1}$$

One common approach (Rosenblatt, 1958) is to search for a pair $(\mathbf{w}, b)$ such that the decision function is given by:

$$f(\mathbf{x}) = \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \tag{3.2}$$

The *weight vector* $\mathbf{w}$ determines the orientation of the decision hyperplane and the *operating point* b determines its distance from the origin. The pair $(\mathbf{w}, b)$ should be chosen so as to minimize the expected risk. Minimizing the risk on the training set is termed *empirical risk minimization* (ERM). This does not necessarily minimize expected risk. By the law of large numbers the empirical risk of a function converges to its expected risk. That is,

$$R_{\mathrm{emp}}(f) \to R(f) \text{ as } l \to \infty. \tag{3.3}$$

This does not imply that for the function f^l minimizing the empirical risk and the function f^{opt} minimizing R the following holds:

$$R_{\mathrm{emp}}(f^l) \to R(f^{\mathrm{opt}}),\ R(f^l) \to R(f^{\mathrm{opt}}). \tag{3.4}$$

That is, there is no guarantee that the empirical risk of the ERM solution converges to the true risk of the optimal solution, or that the true risk of the ERM solution converges to the true risk of the optimal solution (Vapnik, 1998). If the conditions (3.4) do hold then the ERM principle is said to be *consistent*. For consistency of the ERM principle it is necessary to limit the number of decision functions that can be implemented by the learning machine.

The following example, due to Schölkopf et al. (1999a), illustrates why this restriction is necessary. Consider a learning machine that can implement *all* functions from $\Re^d$ to $\{-1, +1\}$. Given a training set (3.1) and test set,

$$\{\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_{\overline{l}}\},\ \overline{\mathbf{x}}_j \in \Re^d, \tag{3.5}$$

such that:

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \cap \{\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_{\overline{l}}\} = \emptyset, \tag{3.6}$$

for any function f there exists a function f^* such that:

$$f^*(\mathbf{x}_i) = f(\mathbf{x}_i)\forall i, \tag{3.7}$$

$$f^*(\bar{\mathbf{x}}_j) \neq f(\bar{\mathbf{x}}_j)\forall j. \tag{3.8}$$

That is, based on the training set there is no way to distinguish the two functions. On the test set they make opposite predictions. This is an extreme example of the phenomenon of *overfitting*. That is, the decision hyperplane is faithful to the training data to the extent of capturing idiosyncracies caused by noisy or irrelevant measurements. Such a decision hyperplane will not lead to good predictions on unseen data with differing idiosyncracies.

Vapnik-Chervonenkis (VC) theory (Vapnik and Chervonenkis, 1974) provides ways to measure the size of the hypothesis space. A *separating hyperplane* with respect to the set S (3.1) is a pair $(\mathbf{w}, b)$ such that $(\forall i)(y_i f(\mathbf{x}_i) > 0)$, i.e. all of the training points are correctly classified. The *margin* of an hyperplane is the distance from the hyperplane to the nearest training point. A necessary and sufficient condition for consistency of ERM is that the classification margin is large with respect to the scale of the data. The *optimal separating hyperplane* is that separating hyperplane such that the distance from $\{\mathbf{x}|\ \langle \mathbf{w}, \mathbf{x}\rangle + b = 0\}$ to the nearest training point is maximal (Vapnik and Lerner, 1963). To eliminate the scaling freedom in the definition of the hyperplane the following definition is made (Vapnik, 1998). An hyperplane is said to be in *canonical* form with respect to the training data (3.1) if $\min_{\mathbf{x}_i \in \Re^d} |\langle \mathbf{w}, \mathbf{x}_i\rangle + b| = 1$. For canonical hyperplanes the margin is:

$$\min_{\mathbf{x}_i \in \Re^d} \left| \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}_i \right\rangle + \frac{b}{\|\mathbf{w}\|} \right| = \frac{1}{\|\mathbf{w}\|}. \tag{3.9}$$

By maximizing the margin it is possible to ensure that the difference between the empirical risk and the true risk is not large, and tends to zero as $l \to \infty$. Thus it is possible to learn hyperplanes in very high dimensional spaces without overfitting the training data.

3.1.2 Margin Maximization as a Quadratic Program

The SVM learns an hyperplane in the feature space $\mathcal{H}$ defined by the mapping:

$$\mathcal{X} \to \mathcal{H},$$
$$\mathbf{x} \to \psi(\mathbf{x}).$$

The mapping ψ need not be explicitly constructed as it only occurs in the training algorithm and decision function in terms of its inner product, which is defined by the *kernel function*:

$$K(\mathbf{x}, \mathbf{z}) = \langle \psi(\mathbf{x}), \psi(\mathbf{z})\rangle_{\mathcal{H}}. \tag{3.10}$$

Any continuous symmetric kernel of a positive definite integral operator on $L^2(\mathcal{X})$, where $\mathcal{X}$ is some compact space, corresponds to an inner product in some feature space (Mercer, 1909). It is usual to specify the kernel function as opposed to the mapping as the kernel function can be constructed based on domain knowledge (Zien et al., 2000). Alternatively one can generate a known classifier structure such as a Gaussian RBF network or two-layer perceptron (Osuna et al., 1997b). Two

commonly used kernels are the *linear kernel*,

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle = \sum_{p=1}^{d} x_p z_p, \tag{3.11}$$

and the *Gaussian kernel*,

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right). \tag{3.12}$$

The linear kernel (3.11) learns an hyperplane in the input space. The Gaussian kernel (3.12) learns an hyperplane in a countably infinite dimensional space. As will be seen below, this corresponds to the decision function (2.6) used above in Chapter 2 (Section 2.3.5.2), i.e. that of an RBF network.

As described in Chapter 1 (Section 1.3) and Section 3.1.1 above, if the hypothesis class is very large then there is the danger of overfitting. The SVM method avoids overfitting by maximizing the margin between the two classes of the training data. This is equivalent to minimizing the norm of the weight vector, $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$ by (3.9). This is achieved by solving the quadratic program (QP):

$$\underset{\mathbf{w},b}{\text{Minimize}} \qquad \Phi(\mathbf{w}, b) = \tfrac{1}{2}\|\mathbf{w}\|^2, \tag{3.13}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle + b) \geq 1, \; i = 1, \ldots, l. \tag{3.14}$$

The constraints (3.14) enforce separability of the training data. To allow for training errors a vector of slack variables $\xi = (\xi_1, \ldots, \xi_l)^T$ is introduced, which yields the following QP (Burges, 1998):

$$\underset{\mathbf{w},b,\xi}{\text{Minimize}} \qquad \Phi(\mathbf{w}, \xi) = \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_{i=1}^{l} \xi_i^k, \tag{3.15}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \; \xi_i \geq 0, \; i = 1, \ldots, l, \tag{3.16}$$

where C is a regularization parameter that must be specified beforehand. The slack variables ξ_i measure the violation of the separability constraints (3.14) and k defines the norm on such errors. If $k = 0$ then solving the quadratic program amounts to minimizing the number of errors while maximizing the margin. This is an NP-complete problem (Vapnik, 1998). The lowest value of k for which the QP (3.15) is tractable is $k = 1$. Although the case $k = 2$ is also tractable it is likely to be more sensitive to outliers (Huber, 1981). An experimental comparison (Mangasarian and Musicant, 2000) shows there to be little difference in practice. In the following the value $k = 1$ is used. The QP (3.15) is termed the *soft margin* formulation of the SVM, since margin violations are allowed.

The QP (3.15) is said to the in the *primal* form. It is normally formulated in the Wolfe *dual* (Cristianini and Shawe-Taylor, 2000). This allows computation in the feature space $\mathcal{H}$ to performed using only inner products $K(\mathbf{x}, \mathbf{z}) = \langle \psi(\mathbf{x}), \psi(\mathbf{z}) \rangle_{\mathcal{H}}$. The solution to the QP (3.15) is found by maximizing

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\|\mathbf{w}\|^2 = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{3.17}$$

subject to

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \; C \geq \alpha_i \geq 0, \; i = 1, \ldots, l. \tag{3.18}$$

The coefficients α_i are the Lagrange multipliers associated with the constraints (3.16). At the maximum $\partial W/\partial\alpha = 0$ so that the weight vector is given by:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i^* y_i \psi(\mathbf{x}_i), \tag{3.19}$$

where $\alpha^* = (\alpha_1^*, \ldots, \alpha_l^*)^T$ is the maximizer of (3.17). The operating point b^* is given by (Burges and Crisp, 2000):

$$b^* = \max\left\{ \max_{y_i=-1, \alpha_i \neq 0} (-1 - \langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle_{\mathcal{H}}), \max_{y_i=+1, \alpha_i=0} (1 - \langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle_{\mathcal{H}}) \right\}. \tag{3.20}$$

This yields the decision function:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{H}} + b^*\right) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^*\right). \tag{3.21}$$

Note that at the maximum of (3.17) many of the Lagrange multipliers α_i may be zero. Thus the decision function (3.21) will be an expansion on a subset of the data. These points are known as *support vectors*. A solution that is an expansion on a small subset of the data is desirable since it aids model interpretation — the support vectors are in some sense the most informative patterns in the data set — and reduces evaluation time. The support vectors form a compression scheme for the data set (Littlestone and Warmuth, 1986) and so minimizing the number of support vectors should lead to better generalization (see Section 3.1.5.2 below).

It is possible that enforcing a low training error, by setting C high in (3.15), will lead to a poorer generalization accuracy. Maximizing the margin w.r.t. all training points can lead to a small margin. If some points are allowed to be *margin errors*, that is lie within the margin band ($\xi_i > 0$), then a larger margin can be achieved w.r.t. the other training points. Conversely, if C is too small then little emphasis is placed on minimizing the training error and an oversmoothed hypothesis will be returned. Hence it is possible to underfit the training data by setting C to be too small, or overfit by fitting C too high. This is illustrated in Figure 3.1 for the Diabetes dataset (see Section 3.1.5.3 below for details). The solution tends to be fairly insensitive to the constant C for Gaussian kernels. It is much more sensitive to the width parameter σ. The sensitivity of the SVM to σ is investigated below in Section 3.1.5.3.

The kernel actually plays two roles. Firstly, it allows computation to be carried out in the high-dimensional feature space. Secondly, it acts as a regularizer, the type of regularization depending on the kernel (Mangasarian, 2000; Williamson et al., 1999). For a more detailed discussion on the nature of the regularization performed by SVMs see Smola et al. (1998) and Evgeniou (2000).

3.1.3 Implementation

Once a solution has been obtained, it may be desired to retrain with additional data, or to retrain with some data removed (e.g. incorrect examples). In a cross-validation exercise, many of the training points remain the same from one optimization to the next. In these cases, it is desirable to start retraining from the previous solution α^*. For example, in the case of a leave-one-out procedure, if example $(\mathbf{x}_i, y_i)$ is left out then the initial vector of Lagrange multipliers is taken to be $(\alpha_1^*, \ldots, \alpha_{i-1}^*, \alpha_{i+1}^*, \ldots, \alpha_l^*)$. When maximizing the Wolfe dual (3.17) it is necessary to enforce the constraint $\sum_{i=1}^{l} y_i \alpha_i = 0$. Thus this initial vector suggested will not be feasible. If the primal

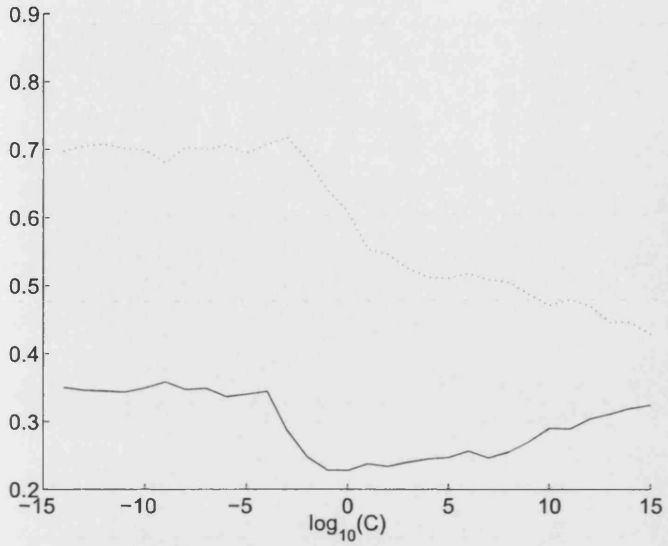

Figure 3.1: Regularization: Diabetes. The test error (*solid line*) has a minimum at around $C = 1$. The proportion of training examples that become support vectors (*dotted line*) decreases as C increases. When C is too small the SVM underfits the training data, when C is too large then the SVM overfits the training data. The SVM was trained with a Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-\|x - z\|^2/d)$. Results are averaged over ten partitions of the data into training and test sets of equal sizes.

formulation (3.15) with $k = 1$ is modified to:

$$\underset{\mathbf{w},b,\xi}{\text{Minimize}} \qquad \Phi(\mathbf{w}, b, \xi) = \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{1}{2}b^2 + C\textstyle\sum_{i=1}^{l} \xi_i, \tag{3.22}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle_{\mathcal{H}} + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, \ldots, l, \tag{3.23}$$

then this constraint need not be enforced. An initial vector such as $(\alpha_1^*, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_l^*)$ would then be feasible. Additionally, this QP is more easily implementable. In this case the operating point is given by:

$$b^* = \sum_{i=1}^{l} y_i \alpha_i. \tag{3.24}$$

Solving the QP (3.22) is equivalent to learning an hyperplane in the space generated by the mapping $\widetilde{\psi}(\mathbf{x}) = (\psi(\mathbf{x}), 1)$ (Cristianini and Shawe-Taylor, 2000). This is analogous to having an input unit in a neural network that has a fixed input of 1 and a weight of $w_0 = b$ (Bishop, 1995). Theoretically, the performance of the classifier given by (3.22) is slightly worse than that given by (3.15) (Cristianini and Shawe-Taylor, 2000). In practice the performance of the two learning machines is the same (Hsu and Lin, 2002).

Standard QP packages are not well suited to the QP arising in the SVM formulation (Frieß et al., 1998). The QP associated with the SVM has l Lagrange multipliers and the number of inner products, $K(\mathbf{x}_i, \mathbf{x}_j)$, is $l \times l$. The QP (3.22) is usually solved in its dual formulation (3.17) by a decomposition routine (Osuna et al., 1997a; Joachims, 1999a; Hsu and Lin, 2002). A subset of the data of size q is chosen at each iteration. This subset is termed the *working set*. The Lagrange multipliers for this subset are then optimized whilst keeping the remainder of them fixed. This

procedure is repeated until convergence. The advantages of this method are that the memory constraints are reduced — by caching kernel calculations $K(\mathbf{x}_i, \mathbf{x}_j)$ during training as opposed to evaluating the entire matrix beforehand — and training time is reduced. A further advantage, exploited in Chapter 4 and Joachims (1999a), is that, at each iteration, heuristics can be applied to accelerate training, tune the model parameters, such as C and σ, or reduce the model complexity.

An hyperplane $(\mathbf{w}, b)$ is an optimally separating hyperplane, if and only if the corresponding α and ξ values are a global optimum to the QP (3.22). This is the case if and only if the Karush Kuhn-Tucker (KKT) criteria are met (Vapnik, 1998). The KKT conditions for the QP (3.22) are equivalent to (Hsu and Lin, 2002):

$$\begin{aligned} -\epsilon \leq ((Q + yy^T)\alpha)_i - 1 \leq \epsilon, & \quad \text{if } 0 < \alpha_i < C, \\ ((Q + yy^T)\alpha)_i - 1 \geq -\epsilon, & \quad \text{if } \alpha_i = 0, \\ ((Q + yy^T)\alpha)_i - 1 \leq \epsilon, & \quad \text{if } \alpha_i = C, \end{aligned} \tag{3.25}$$

where $Q_{ij} = \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$, $y = (y_1, \ldots, y_l)^T$ and $\alpha = (\alpha_i, \ldots, \alpha_l)^T$. Theoretically the KKT conditions require $\epsilon = 0$, in practice a looser stopping criterion of $\epsilon = 0.001$ is used (Joachims, 1999a; Hsu and Lin, 2002). The solutions obtained by enforcing a stricter stopping criterion are rarely different from those obtained with $\epsilon = 0.001$.

Unless stated otherwise, the SVM implementation used in this work uses the working set selection strategy of Hsu and Lin (2002), implemented in Matlab 6.0 (Matlab 6.0, 2001). The working set size is set at $q = 10$. The subproblem optimization on the working sets is performed using the method of Hildreth (1957). The termination precision of the subproblem optimization is machine accuracy (10^{-16}) and the precison of the termination criteria (3.25) is $\epsilon = 0.001$.

3.1.4 Variable Misclassification Costs

In the optimization problem (3.22) all of the errors ξ_i have equal weight C in the minimand. That is, the cost $\lambda(+1, -1)$ of mislabelling a negative example as a positive one is the same as the cost $\lambda(-1, +1)$ of mislabelling a positive example as negative. When there are unequal misclassification costs it is necessary to place different weights on the errors. It may also be known that the proportion of positives π^s_{+1} in the training sample is different to the proportion π_{+1} of unseen examples that we wish to classify. This may be the case in drug discovery. For example, a set of compounds may have been chosen for screening since they were expected to show some activity, whereas the proportion of actives in general is low. Conversely, a set of compounds whose activity it is desired to predict may be have been chosen on the basis of compounds already screened, and are likely to exhibit higher activity. In either case it is necessary to reweight the errors in the optimization.

Lin et al. (2000) show that the term $C \sum_{i=1}^{l} \xi_i$ in (3.22) should be split into $C^+ \sum_{i|y_i=+1} \xi_i + C^- \sum_{i|y_i=-1} \xi_i$. The regularization constants C^+ and C^- are then given by:

$$C^+ = \lambda(-1, +1)\pi^s_{+1}\pi_{-1}C, \tag{3.26}$$

$$C^- = \lambda(+1, -1)\pi^s_{-1}\pi_{+1}C, \tag{3.27}$$

where $\pi^s_{\pm 1}$ are the proportions of positive and negative examples in the training sample and $\pi_{\pm 1}$ are the 'true' proportions (or the proportions for the set of examples on which we wish to make predictions)[1]. It is still only necessary to specify one regularization parameter C, which controls

[1]Which could be estimated from previous experience.

the overall amount of regularization.

3.1.5 Bounds on the Expected Error

The SVM maximizes the margin w.r.t. those points with $\xi_i = 0$ whilst minimizing $\sum_{i=1}^{l} \xi_i$. Margin maximization is an example of *capacity control*, that is, restricting the 'size' of the hypothesis space. The 'size' of an hypothesis space can be quantified by its Vapnik-Chervonenkis (VC) dimension. A set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ is said to be *shattered* by the set of hypotheses $\mathcal{F}$ if for any labelling $\{y_1, \ldots, y_l\} \in \{-1, +1\}^l$ there exists a function $f \in \mathcal{F}$ such that $f(x_i) = y_i$, $i = 1, \ldots, l$. For example, hyperplanes $f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ in $\Re^d$ can shatter any set of $d+1$ points, in general position. The *VC dimension*, h, of a set of hypotheses $\mathcal{F}$ is defined to be the size of the largest set of points that can be shattered by the set $\mathcal{F}$. There is no set of $d+2$ points in $\Re^d$ that can be shattered by hyperplanes. Hence, the VC dimension of hyperplanes in $\Re^d$ is $d+1$. This is also the number of free parameters of an hyperplane. This relationship between VC dimension and the number of parameters does not generally hold for more complicated classifiers. VC theory provides bounds on the expected generalization error of a classifier.

An alternative estimate of the expected generalization error of a classifier is the leave-one-out error. The leave-one-out (LOO) error of a classifier is calculated by training on $l-1$ points and predicting the label of the remaining point. This is repeated for all l points and the errors averaged. The expected leave-one-out error is an almost unbiased estimate of the expected generalization error (Vapnik, 1998). The leave-one-out error is computationally expensive to compute. For support vector machines, the leave-one-out error can be upper bounded by analysing the geometry of the support vectors (Chapelle et al., 2002).

These bounds on the generalization error and LOO error avoid the need to reserve a portion of the data as a validation set for model selection, or to use computationally expensive cross-validation.

3.1.5.1 Bounds from VC-Theory

If the training data (3.1) are linearly separable in the feature space $\mathcal{H}$ then the following bound on the expected generalization error ε holds (Vapnik, 1998; Shawe-Taylor et al., 1996):

$$\varepsilon \leq O\left(\frac{R^2}{l\gamma^2}\right), \tag{3.28}$$

where R is the radius of the smallest ball in feature space containing the training data, l is the number of training examples and γ is the margin. The quantity R^2/γ^2 is an estimate of the VC-dimension of the set of hyperplanes separating the data with margin γ.

The radius R can be found by solving a quadratic programme of size l (Burges, 1998). This additional computational time can be avoided in certain cases. When using a linear kernel, the training examples can be normalized to have length 1. For high dimensional data, e.g. in document classification (Joachims, 2000), this reduction in dimensionality does not affect the performance. When using a radial basis function kernel, the input data are mapped to the surface of the unit sphere in feature space, since $\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_{\mathcal{H}} = \exp(-\|\mathbf{x} - \mathbf{x}\|/2\sigma^2) = 1$. Hence, one can take $R = 1$ as an upper bound on the radius of the smallest enclosing sphere. If σ is large then the data tend to be mapped to a very flat ellipsoid in feature space (Schölkopf et al., 1999b), the approximation $R = 1$ then leads to a loose bound. To obtain a better estimate of the generalization error the data can be rescaled in feature space (Chapelle and Vapnik, 2000) using the method

of kernel principal components analysis (Schölkopf et al., 1997a), although this requires further computational expense.

For the case where the errors ξ are penalized linearly, i.e. $k = 1$ in (3.22), this bound is only applicable when the data are separable. For all of the data sets used in this thesis, enforcing separability leads to a poorer solution than does allowing errors ($\xi > 1$) or margin errors ($\xi > 0$). Hence, this bound was not used in the work presented in this thesis. For the case where the errors are penalized quadratically ($k = 2$) this bound can be used for separable and non-separable problems, and has been shown to be predictive on a number of data sets (Duan et al., 2001; Chapelle et al., 2002).

3.1.5.2 Bounds on the Leave-One-Out Error

An alternative estimate of the expected error is the leave-one-out error. The *leave-one-out* (LOO) error of a classifier with respect to a training set S (3.1) is:

$$\frac{1}{l}\mathcal{L}(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l) = \frac{1}{l}\sum_{i=1}^{l}\delta(f_{S\setminus i}(\mathbf{x}_i), y_i), \tag{3.29}$$

where $f_{S\setminus i}$ is the classifier trained on the set S with the example $(\mathbf{x}_i, y_i)$ removed, and δ is the Kronecker delta. The leave-one-out error is an almost unbiased estimate of the expected generalization error (Vapnik, 1998), that is,

$$E(\varepsilon^{l-1}) = \frac{1}{l}E(\mathcal{L}(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l)), \tag{3.30}$$

where ε^{l-1} is the generalization error of a classifier trained on $l-1$ examples and the expectations are taken over random choice of sample. The leave-one-out error is expensive to compute. For an SVM it is possible to obtain an upper bound on this quantity. Since, on removing a non-support vector from the training set, the solution computed does not change, the non-support vectors will not be misclassified in the leave-one-out procedure. Hence (Vapnik, 1995),

$$\mathcal{L}(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l) \leq N_{SV}, \tag{3.31}$$

where N_{SV} is the number of support vectors. Tighter bounds can be achieved by analysing the geometry of the support vectors (Chapelle et al., 2002). Vapnik and Chapelle (2000) provide a quantity based on the concept of the *span* of a support vector. Under the assumption that the set of support vectors remains unchanged during the leave-one-out procedure this gives the exact number of errors made during the leave-one-out procedure. This assumption is not likely to be true in practice, but empirical evidence (Chapelle et al., 2002; Chapelle and Vapnik, 2000) suggests that this bound is predictive and tight. The authors suggest using the VC bound (3.28) in practice, however, as calculation of the span bound requires approximately as much computational time as training the SVM. During parameter selection typically many models are built and need to be rapidly evaluated. The VC bound is only valid for separable data if the errors are linearly penalized ($k = 1$ in (3.22)). The VC bound (3.28) can be used for non-separable data if $k = 2$. This formulation is not used here.

A computationally efficient estimate of the leave-one-out error, related to the span bound, is given by the following.

Definition 3.1 *The $\alpha\xi$ estimator of generalization error is given by:*

$$\varepsilon^l_{\alpha\xi} = \frac{1}{l}\#\{i : \rho\alpha_i R^2 + \xi_i \geq 1\}, \tag{3.32}$$

where $\#$ denotes cardinality, $\rho = 2$ and R^2 is an upper bound on $\max_i(K(\mathbf{x}_i, \mathbf{x}_i))$, i.e. the radius of a sphere centred on the origin in feature space containing all of the training examples.

The $\alpha\xi$ estimator is an upper bound on the leave-one-out error.

Theorem 3.1 (Joachims, 2000) *Provided $(\exists\alpha_i)(\alpha_i \neq C)$, the leave-one-out error of an SVM on a training set S is bounded as:*

$$\frac{1}{l}\mathcal{L}(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l) \leq \varepsilon^l_{\alpha\xi}. \tag{3.33}$$

If $(\exists\alpha_i)(\alpha_i \neq C)$ then the solution is said to be *stable.* There is no guarantee that this is the case, although it is usually true in practice (Burges, 1998). If R is taken to be 1, as in the special cases mentioned above (Section 3.1.5.1) then $\varepsilon^l_{\alpha\xi}$ can be computed at little extra cost from the SVM solution.

When there are unequal misclassification costs, the leave-one-out estimator of risk is defined analogously to (3.29) as:

$$\begin{aligned}
&\frac{1}{l}\mathcal{L}_\lambda(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l) = \\
&\quad \frac{1}{l}\sum_{i=1}^{l}\lambda(f_{S\setminus i}(\mathbf{x}_i), y_i) = \\
&\quad \frac{1}{l}\left\{\sum_{i:y_i=+1}\lambda(+1,-1)FP_{S\setminus i} + \sum_{i:y_i=-1}\lambda(-1,+1)FN_{S\setminus i}\right\},
\end{aligned} \tag{3.34}$$

where λ is the loss function and $FP_{S\setminus i}$ and $FN_{S\setminus i}$ are the number of negative and positive points, respectively, misclassified by the classifier trained on the set $S \setminus i$. It follows from the theorem of Lunts and Brailovskiy (1967) that the leave-one-out estimator of risk (3.35) is an almost unbiased estimator of the true risk.

Definition 3.2 *The $\alpha\xi$ estimator of risk is given by:*

$$\begin{aligned}
\varepsilon^l_{\alpha\xi} &= \frac{1}{P}\#\{i : y_i = +1,\ \rho\alpha_i R^2 + \xi_i \geq 1\}\lambda(-1,+1) \\
&+ \frac{1}{N}\#\{i : y_i = -1,\ \rho\alpha_i R^2 + \xi_i \geq 1\}\lambda(+1,-1),
\end{aligned} \tag{3.35}$$

where R^2 is an upper bound on $\max_i(K(\mathbf{x}_i, \mathbf{x}_i))$, and $\rho = 2$, as before, and P (resp. N) is the number of positive (resp. negative) training examples.

A corollary of Theorem 3.1 is the following.

Corollary 3.2 *Provided $(\exists\alpha_i)(\alpha_i \neq C)$, the leave-one-out estimator of risk of an SVM on a training set S is bounded as:*

$$\frac{1}{l}\mathcal{L}_\lambda(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_l, y_l) \leq \varepsilon^l_{\alpha\xi}. \tag{3.36}$$

This formulation will be useful for model order selection when there are unequal misclassification costs, which is often the case in pharmaceutical data analysis.

(Note that the proof given in Joachims (2000) is for the case where the QP is formulated as (3.15). Since the formulation (3.22) is equivalent to adding a constant to the kernel function and not enforcing the constraint $\alpha^T y = 0$ the proof follows as before.)

3.1.5.3 Empirical Comparison

The results presented above (Section 3.1.5.2) are theoretical in nature. They provide estimates of the expected leave-one-out error or risk, but the following practical questions arise: Do the parameters minimizing the $\alpha\xi$ estimator provide minimal test error? How close to the test error are the estimates? These questions are answered empirically for the problem of choosing the width parameter σ of a Gaussian kernel (3.12) for three data sets. The data sets are as follows.

Cancer This data set was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg (Mangasarian and Wolberg, 1990; Bennett and Mangasarian, 1992). There are 699 examples of tumour cells, characterized by nine physiological measurements. One of the features has 16 missing values, these are replaced by the feature mean. The task is to distinguish benign tumour cells ($y = -1$) from malignant tumour cells ($y = +1$). The proportion of malignant tumour cells is $\pi^s_{+1} = 0.345$.

Diabetes This data set is from the National Institute of Diabetes and Digestive Kidney Diseases (Smith et al., 1998). There are 768 examples of female Pima Indians, characterized by eight physiological descriptors. The task is to predict presence ($y = +1$) or absence ($y = -1$) of diabetes. The proportion of the sample with diabetes is $\pi^s_{+1} = 0.349$.

HTS This data set consists of 489 compounds from the high throughput screening assay a3 (see Section 2.4.2). There are 13 descriptors (feature set fs03). The task is to predict whether a compound is active ($y = +1$) or inactive ($y = -1$). The proportion of actives on this subsample is $\pi^s_{+1} = 0.056$. This is much more unbalanced than the other two data sets.

The first two data sets are available from the UCI Machine Learning Repository (Blake and Merz, 1998). The third is proprietary to GlaxoSmithKline. The attributes of all of the data sets are scaled to lie in the range $[-1, +1]$. This has proven in my experience to result in faster convergence of the QP and is common in the SVM literature. It can be considered to be a 1-norm version of standardizing the empirical distribution to have mean zero and standard deviation one.

Each data set is randomly partitioned into training and test sets of equal size. An SVM was trained with $C = 1$ and Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/2\sigma^2)$ with $\sigma \in \{0.1, 0.2, \ldots, 1.0, 1.2, \ldots, 3.6\}$. The $\alpha\xi$ estimator (3.32) was calculated with $R = 1$. The error on the test set was evaluated and the procedure was repeated for ten random partitions of the data into training and test sets and the results averaged. The estimates and test errors are plotted in Figures 3.2–3.4. The minimizer, $\hat{\sigma}$, of the error estimate has performance close to the minimal on the Cancer and Diabetes data. The error estimate is qualitatively different from the test error for the HTS data set. This set has a low proportion of positives and the low error estiamate occurs when the SVM predicts all compounds to be inactive. This is obviously not desirable in practice as it simply tells the chemist not to bother screening anything.

As all three of these data sets have more positive examples than negative examples an alternative measure of performance is to assign a higher cost to misclassification of positive examples. If the

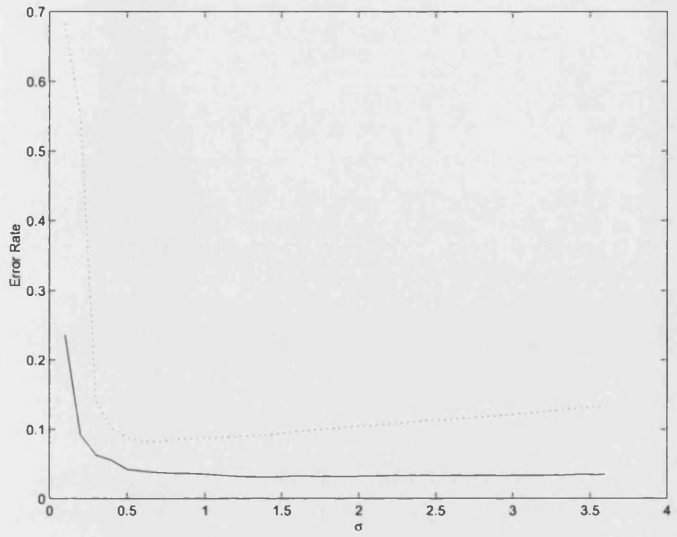

Figure 3.2: Error Bound Comparison: Cancer, $C = 1$. The minimizer, $\hat{\sigma}$, of the $\alpha\xi$ estimate (*dotted line*) underestimates the minimizer, σ^*, of the test error (*solid line*). The test error obtained, $\varepsilon(\hat{\sigma})$, is close to the optimal, $\varepsilon(\sigma^*)$.

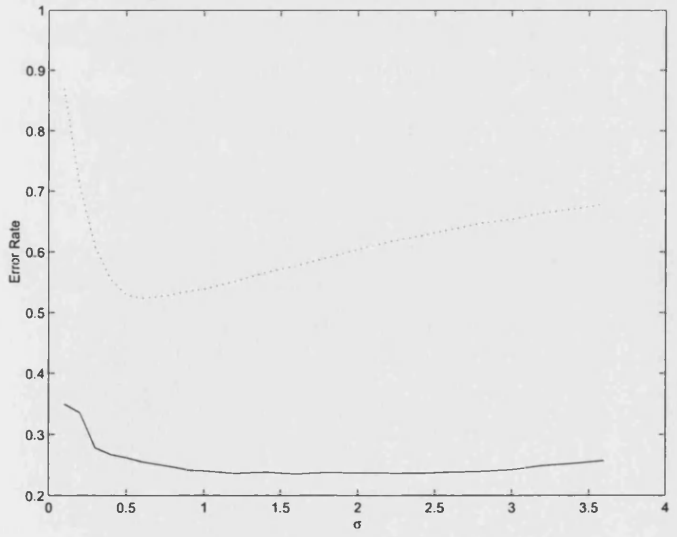

Figure 3.3: Error Bound Comparison: Diabetes, $C = 1$. The test error (*solid line*) attains a minimum at around $\sigma = 1.5$ and increases slightly afterwards. The minimizer of the $\alpha\xi$ estimate (*dotted line*) underestimates the minimizer of the test error.

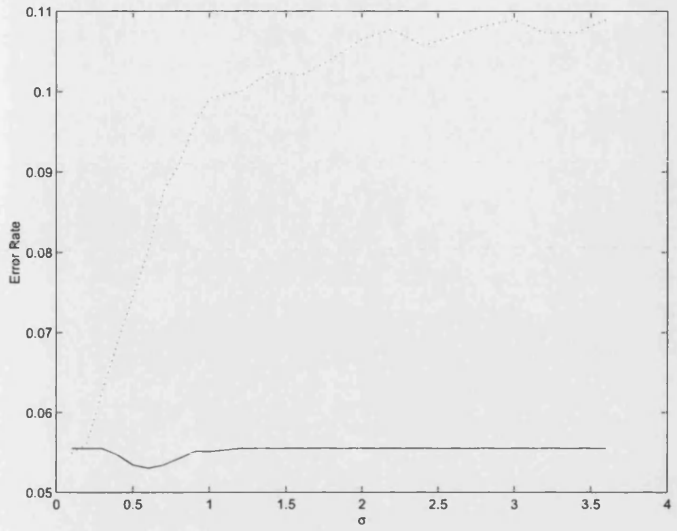

Figure 3.4: Error Bound Comparison: HTS, $C = 1$. The test error (*solid line*) is more-or-less constant over the range of σ tried. The $\alpha\xi$ estimate (*dotted line*) increases with σ. The behaviour of the $\alpha\xi$ estimate is qualitatively different from that of the test error on these data.

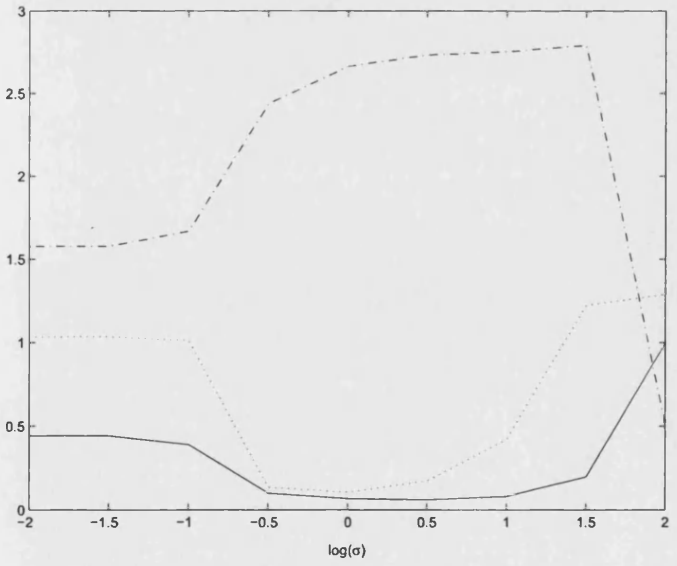

Figure 3.5: Risk Bound Comparison: Cancer, $C = 1$. The minimizer of the $\alpha\xi$ estimate (*dotted line*) is close to the minimizer of the test risk (*solid line*). The relative advantage (*dash-dotted line*) varies inversely with the test risk.

misclassification costs are reweighted according to the class distributions then the cost of a false negative is $\lambda(-1,+1) = 1/\pi_{+1}$ and the cost of a false positive is $\lambda(+1,-1) = 1/\pi_{-1}$.

The risk functional for classification is:

$$\begin{aligned} R(f(\cdot;\mathbf{w},b)) &= \lambda(+1,-1)\pi_{-1}\Pr\{f(\mathbf{x};\mathbf{w},b) = +1|y=-1)\} \\ &+ \lambda(-1,+1)\pi_{+1}\Pr\{f(\mathbf{x};\mathbf{w},b) = -1|y=+1)\}, \end{aligned} \tag{3.37}$$

where π_k is the prior probability of class $k \in \{-1,+1\}$ and $\lambda(l,k)$ is the loss incurred for making prediction l when the true class is k. If we set $\lambda(\mp 1,\pm 1) = 1/\pi_{\pm 1}$ then this becomes:

$$R(f(\cdot;\mathbf{w},b)) = \Pr\{f(\mathbf{x};\mathbf{w},b) = +1|y=-1)\} + \Pr\{f(\mathbf{x};\mathbf{w},b) = -1|y=+1)\}. \tag{3.38}$$

In practice, the training set proportions π_k^s are used instead. Minimizing this risk functional is equivalent to minimizing:

$$\frac{FP}{N} + \frac{FN}{P},$$

instead of:

$$\frac{FP+FN}{N+P},$$

where FP (resp. FN) is the number of false positives (resp. negatives) and P (resp. N) is the number of positives (resp. negatives) in the training set. The two approaches are equivalent for balanced data.

The $\alpha\xi$ estimate of expected risk was calculated by (3.35). The range of σ used above was in a neighborhood of the optimal value. To obtain a broader picture of the performance and the risk estimate, $\log(\sigma)$ was varied in $\{-2,-1.5,\ldots,2\}$. The estimates and test risk are plotted in Figures 3.5–3.7. (The performance and estimates for $\sigma = 0.001$ were identical to those for $\sigma = 0.01$ for all data sets.) Also plotted is the relative advantage. We will return to relative advantage as a performance measure later in Section 3.2.2.

For the first two data sets (Figures 3.5 and 3.6) the $\alpha\xi$ estimate is similar to the test risk. For the HTS data the estimate is not predictive since it is much tighter when the risk is close to 1 than at other values. This corresponds to predicting all negatives as positive, which is not desirable.

In summary, for Gaussian kernels the $\alpha\xi$ estimate is a tight upper bound on the test error, and can be generalized to estimate test risk. The minimizer, $\hat{\sigma}$, of the estimate is close to the minimizer of the risk. Furthermore, the risk obtained by using a value of $\hat{\sigma}$ that minimizes the estimate is close to the risk of using the optimal value. There are two disadvantages in using the $\alpha\xi$ estimate. Firstly, it is not valid when the solution is unstable, i.e. when all of the Lagrange multipliers α_i are bounded at C. Anecdotal evidence suggests that this does not happen very often and that unstable solutions are rarely optimal. Secondly, the estimator is not a reliable predictor near regions in parameter space where the SVM is predicting $y = -1$ (or $y = 1$) for all $\mathbf{x}$. This is more likely to be the case when the proportion of positives or negatives (resp.) is low. On the HTS data, the propotion of positives is low and the $\alpha\xi$ estimator performs poorly. These solutions can be discarded as the SVM will also predict $y_i = 1$ for all $\mathbf{x}_i$ in the training set.

Duan et al. (2001) show that the best estimator of test error for an SVM is given by five-fold cross-validation. However, this is a computationally intensive technique. They compared several easily computable estimators of a Gaussian SVM's performance. They showed that the $\alpha\xi$ estimate and generalized approximate cross-validation (GACV) (Wahba et al., 2000) perform well. However, GACV tends to be biased towards small σ and small C. The other measures

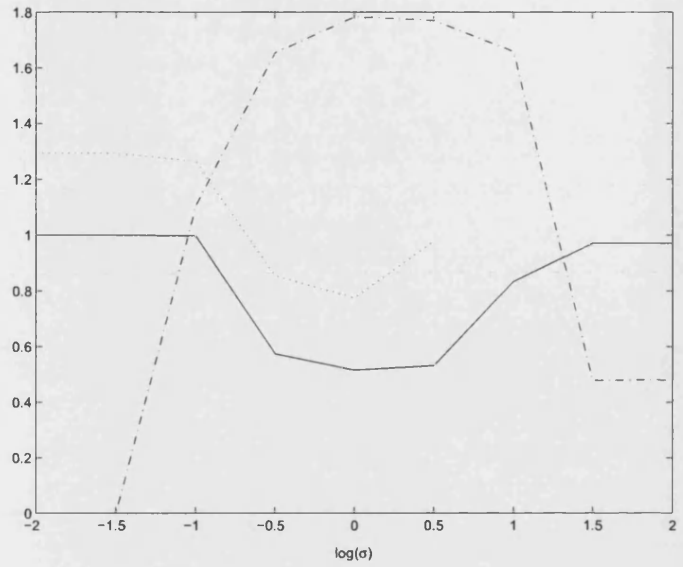

Figure 3.6: Risk Bound Comparison: Diabetes, $C = 1$. The minimizer of the $\alpha\xi$ estimate (*dotted line*) is the minimizer of the test risk (*solid line*). The solution is unstable for $\log(\sigma) > 0.5$ so the error estimate does not hold. The relative advantage (*dash-dotted line*) varies inversely with the test risk.

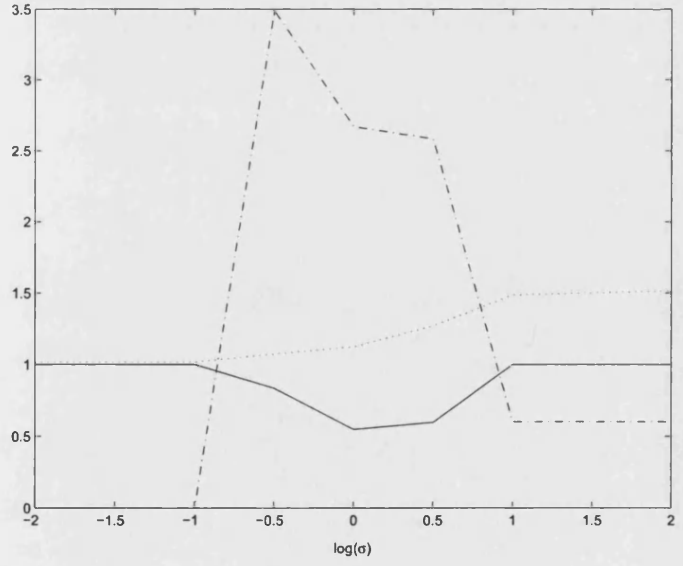

Figure 3.7: Risk Bound Comparison: HTS, $C = 1$. The $\alpha\xi$ estimate (*dotted line*) is tighter when the risk (*solid line*) is 1. This corresponds to predicting all positives or all negatives. The $\alpha\xi$ estimate is qualitatively the same for the other values, although overall it is not predictive since it is too loose. The relative advantage (*dash-dotted line*) varies inversely with the test risk.

evaluated were the radius margin bound (see Section 3.1.5.1), an approximation to the span bound of Chapelle et al. (2002), and a VC bound of Vapnik (1998). All of these performed poorly, yielding bounds qualitatively different to the error and not predictive of the optimal parameters.

3.2 Performance Measures

Before considering the application of SVMs to pharmaceutical data analysis, I describe some performance measures. The first is the ROC curve, it is more informative than reporting a single number such as error rate or risk. The second is relative advantage, which is related to the global enhancement described in Section 2.4.2.1.

3.2.1 Receiver Operating Characteristic Curves

When classifying unseen data two types of mistake are possible: misclassifying a positive example ($y = +1$) as negative ($y = -1$) and misclassifying a negative example as positive. The first type of error is called a false negative (FN) and the second a false positive (FP). We can write:

$$\begin{aligned} \text{total positives}(P) &= \text{true positives}(TP) + \text{false negatives}(FN), \\ \text{total negatives}(N) &= \text{true negatives}(TN) + \text{false positives}(FP). \end{aligned}$$

Thus, the performance of the classifier can be summed up by two ratios: the true positive rate, or hit rate, TP/P and the false positive rate, or false alarm rate, FP/N. Under fixed misclassification costs $\lambda(l,k)$, $l,k \in \{-1,+1\}$ with no rewards, $\lambda(l,l) = 0$, the aim is to minimize the expected risk,

$$\begin{aligned} R(f(\cdot;\mathbf{w},b)) &= \lambda(+1,-1)\pi_{-1}\Pr\{f(\mathbf{x};\mathbf{w},b) = +1|y=-1)\} \\ &+ \lambda(-1,+1)\pi_{+1}\Pr\{f(\mathbf{x};\mathbf{w},b) = -1|y=+1)\}, \end{aligned} \tag{3.39}$$

where π_k is the prior probability of class k, i.e. the class population proportion. Note that:

$$\lambda(+1,-1)\pi_{-1}\frac{FP}{N} + \lambda(-1,+1)\pi_{+1}\left(1 - \frac{TP}{P}\right) \to R(f(\cdot;\mathbf{w},b))$$

as $l \to \infty$. In practice, the true positive and false positive rates and the risk are evaluated on some set of data held out from the training set, known as a test set.

It is now easy to see how a classifier giving high accuracy (say, on the test set) can fail to be of any use. In the drug discovery domain we wish to identify active compounds. In a given set of compounds the proportion of actives may be 5%. In this case a classifier could happily predict that all compounds are inactive and achieve 5% error. That this high level of accuracy is totally useless is apparent from the TP rate of 0 (the FP rate will be 0 also). A classifer with a TP rate of 0.8 and an FP rate of 0.2 will have a higher error of 20% but will correctly identify 4% of the compounds as active. From (3.39) the error is the risk under equal misclassification costs. In order to obtain a useful classifier the misclassification costs must taken into account. When classifying compounds as active or inactive, the cost of a false positive is the cost of subsequently screening that compound *in vitro* (or *in vivo*). The cost of a missed positive is very difficult to quantify, but will vary over different assays, at different stages of the process and according to the opinions of different chemists.

In many domains the misclassification costs are not known *a priori*, or may be variable. In this case it is desirable to have a classifier that performs well over a range of costs. If the cost of a

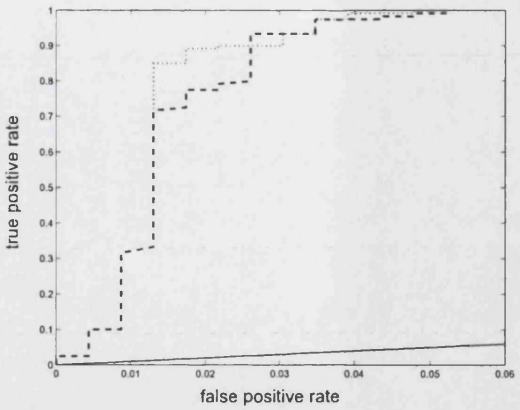

Figure 3.8: ROC curves: Cancer. The ROC curve is generated by varying the decision threshold b for an SVM ($C = 1$). Neither the RBF kernel (*dashed line*) nor the linear kernel (*dotted line*) has an ROC entirely containing the other. Both are much better than random (*solid line*). Note that only a small portion of the ROC curves are shown, as beyond a false positive rate of 0.06 both classifiers have a true positive rate of 1.

false alarm $\lambda(+1,-1)$ is increased then a lower false positive rate is desired. If the cost of a missed positive $\lambda(-1,+1)$ is increased then a higher true positive rate is desired. The true positive rate and false positive rate of an SVM, or of any classifier of the form (3.2) or (3.21)) can be varied for a given $\mathbf{w}$ by varying the parameter b, known as the *operating point*. As $b \to -\infty$ both rates $\to 0$, whilst as $b \to \infty$ both rates $\to 1$. This produces a *receiver operating characteristic (ROC) curve* that gives a picture of the performance of the classifier over a range of costs. The area under the ROC curve ($AUROC$) is a measure of the performance. It is equal to the Wilcoxon statistic, a distribution independent measure of classifier performance (Hanley and McNeil, 1982). A random classifier will have $AUROC = 0.5$ and a classifier that has perfect performance over all costs will have $AUROC = 1$. (If a classifier is worse than random, i.e. $AUROC < 0.5$, then reverse its outputs to obtain $AUROC > 0.5$.) An example ROC curve is shown in Figure 3.8.

Note that for an SVM there is no guarantee that the operating point given by (3.20) or (3.24) is Bayes optimal, i.e. minimizes the true risk (Platt, 2000). Thus it may be advantageous to plot an ROC curve even if the misclassification costs are fixed, in order to optimize the operating point. The Bayes optimal risk, for a given $\mathbf{w}$, is obtained by setting the operating point to be such that the slope of the ROC curve is $\lambda(+1,-1)/\lambda(-1,+1)$.

Friedman (1997) decomposes expected classification error rate into 'boundary-bias' and 'variance' and shows that intuition derived from the bias-variance trade-off in function estimation does not carry over to classification. He suggests, and supports with empirical evidence, that optimizing the operating point can reduce error rate (or risk) for biased learning algorithms such as k-nearest neighbour (Mitchell, 1997) and naïve Bayes (Duda and Hart, 1973), and aggregated methods such as bagging (Breiman, 1994). For unequal class priors the correction to the operating point is achieved by reweighting the examples according to the priors. This is equivalent to specifying the cost function of Section 3.1.5.3 above. Even when the priors are equal, the operating point can be optimized. Feng (2001) shows that the operating point of the standard SVM is suboptimal when the classes are asymmetric.

The ROC curve of a classifier allows us to calculate the empirical risk for any specification of costs. It allows the user to set the optimal operating point for a range of criteria including Neyman-Pearson, minimum recall, workforce utilization, etc. (Provost and Fawcett, 2000). It is

thus a more useful indicator of performance than simply reporting error rates, or true and false positive rates, at a given operating point.

An alternative, but equivalent, summary of the performance is to consider the trade-off between precision and recall. Precision is the probability that a compound classified as positive truly is positive and recall is the probability that a positive example is correctly classified as such. These quantities are estimated as:

$$\text{precision} = \Pr\{y = +1 | f(\mathbf{x}; \mathbf{w}, b) = +1\} = \frac{TP}{TP + FP}, \tag{3.40}$$

$$\text{recall} = \Pr\{f(\mathbf{x}; \mathbf{w}, b) = +1 | y = +1\} = \frac{TP}{P}. \tag{3.41}$$

The precision-recall break-even point is defined to be that value for which the recall and precision are equal, i.e. for which the number of false positives equals the number of false negatives. This is a commonly used measure of performance in text classification (Joachims, 1998) where documents may belong to more than one category. These measures are also useful in drug design since a compound may be active against no targets, one target, or many targets.

3.2.2 Relative Advantage

In Chapter 2 (Section 2.4.2.1) above, the global enhancement was introduced as a useful measure of performance when analysing screening data. This measure was derived from the $A50$ and thus a ranking of compounds in order of activity is required to calculate it. A related measure is outlined below, which does not require such a ranking.

Define the AQ of a predictive technique to be the number of compounds it is necessary to screen to find $Q\%$ of the actives. Recall the definitions of Section 3.2.1. For a classifier, the percentage of actives found is $Q = TP/P \times 100$. The number that must be screened to find these actives is the number predicted positive, viz.:

$$AQ = TP + FP. \tag{3.42}$$

For a random selection:

$$AQ = \frac{TP}{P}(P + N). \tag{3.43}$$

The global enhancement with respect to AQ is defined as the ratio of these quantities,

$$GE = \frac{TP/(TP + FP)}{P/(P + N)} = \frac{\text{precision}}{\pi_{+1}}. \tag{3.44}$$

This is equivalent to the definition of *relative advantage* (RA) proposed in Muggleton et al. (2000). The authors state that this measure of performance is useful when the number of positives is small, and when there is no guarantee that the all of the positives can be identified. This is generally the case when analysing HTS data. The relative advantage is related to the precision by (3.44), when the precision = 1, RA is maximized and equals $1/\pi_{+1}$. RA is not a particularly meaningful measure of performance unless $TP \gg 0$. Dividing the precision by the proportion of active compounds allows a comparison across screens.

It is not possible to maximize relative advantage by minimizing a risk functional of the form (3.39). For a given assay, π_{+1} is fixed, so maximizing relative advantage is equivalent to maximizing the precision, i.e. minimizing:

$$\frac{TP + FP}{TP} = \frac{FP}{TP} + 1. \tag{3.45}$$

Multiply the RHS of (3.45) by the constant P/N and use the binomial expansion to obtain

$$\text{argmax}(RA) = \text{argmin}\left(\frac{FP}{N} + \lambda\frac{FN}{P} + O\left(\left(\frac{FN}{P}\right)^2\right)\right), \tag{3.46}$$

where $\lambda = FP/N$ is the (unknown) false positive ratio. Setting $\lambda = 1$ assumes $TN = 0$ and so errs in favour of minimizing the error rate on the positives (active compounds). This leads to the risk functional already encountered in Sections 3.1.5.3 and 3.3.2. Recall from Figures 3.5–3.7 that minimizing this risk functional corresponds empirically to maximizing relative advantage.

3.3 Applications

The support vector machine has shown good performance on many classification problems including handwritten digit recognition (Cortes and Vapnik, 1995), document classification (Joachims, 1998) and face detection (Osuna et al., 1997b). In Section 3.3.1, I present results on the ordinal regression problem of Section 2.4.1. This work has appeared in Burbidge et al. (2000, 2001a). In Section 3.3.2, I present results on a problem of predicting the activity of competitive antagonists. In Section 3.3.3, I return to the problem of ranking compounds for high-throughput screening.

3.3.1 Predicting the Inhibition of Dihydrofolate Reductase

Recall the problem of predicting the reduction of dihydrofolate reductase from Section 2.4.1. It was shown that a multilayer perceptron achieved a cross-validated error of 0.1381% (s.e.: 0.0090), outperforming a radial basis function network and a decision tree. This improvement in accuracy was at the expense of longer training time. The neural network minimizing expected generalization error was found by time-consuming manual tuning. This model tuning required an order of magnitude more computational time than the training itself. The neural network algorithms in CLEMENTINE that attempt to learn the network structure from the data required correspondingly less computational time, at the expense of performance.

As described in Burbidge et al. (2001a), an SVM was used to classify these data. The SVMs used in that experiment were trained using the SVM^{light} package described in Joachims (1999a), on a SunOS Ultra-2 with 2 processors. Thus, a strict time comparision was not possible as CLEMENTINE was run on an NT workstation. Recently SVM^{light} has been ported to NT, the results reported below are for the same platform as that on which CLEMENTINE was run.

The kernel function used was a Gaussian RBF,

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right), \tag{3.47}$$

which projects the data into a countably infinite dimensional feature space. The data are separable in this space for a suitable value of σ, but, to avoid overfitting, the soft margin formulation of the SVM is used. This leaves two free parameters to be specified, σ and C. These were selected by training an SVM for various values of these parameters, and selecting that pair which minimized the $\alpha\xi$ estimator (3.33). The values tried were $\sigma \in \{\sigma_0, 2\sigma_0, 3\sigma_0\}$ and $C \in \{C_0, 10C_0, 100C_0\}$, where σ_0 is given by a heuristic of Jaakkola et al. (1999) (see also Chapter 4, Section 4.2); and, $C_0 = R^{-2}$ where R is the radius of the smallest ball in feature space containing all of the data, as suggested by Cristianini and Shawe-Taylor (2000). Note that model selection is much easier for an SVM than for neural network algorithms. There are no learning rates or decay terms, and the

free parameters can be set without use of a hold-out validation set.

The cross-validated error of the SVM was 0.1269% (s.e.: 0.0086), which is lower, but not significantly so (again, using the t-test at 95% confidence), than the manually tuned and 'dynamic' neural networks. It is significantly lower than the other algorithms tested. The main advantage of the SVM in this case is that much less human effort was required during model selection. The compuational time was also significantly reduced. The mean training time was 29s, which is an order of magnitude less than for the manually tuned neural network. The total model selection time for the SVM was 596s, compared to 5000–20000s for the various neural networks. Note also, that the model selection time for the SVM includes the training time, as the model selection is performed using all of the available training data. The decision tree was quickest to train, but gave poorer generalization accuracy.

It is interesting to note that an SVM with a Gaussian RBF kernel performed well, yet CLEMENTINE's RBF network performed worst of all of the algorithms tested, even though the number of RBF centres was varied from 2 up to the number of support vectors in the SVM solution (about 200). This suggests either that the k-means clustering method of CLEMENTINE is not as good at locating centres as the SVM, or that a Gaussian RBF is not a good choice for these data. It has been suggested that perhaps the data are poorly distributed and hence a large number of RBF centres are required[2]. An RBF network with a large number of centres is likely to overfit (Bishop, 1995), whereas an SVM, even with many centres (i.e. support vectors) avoids overfitting by means of margin maximization. A comparison of RBF networks and SVMs is provided by Schölkopf et al. (1997b). In that study the improved performance of the SVM appeared to be due to the location of the RBF centres. It is not possible to specify the location the RBF centres in CLEMENTINE, so this was not investigated. It remains an avenue for further research.

3.3.2 Predicting Activity of Competitive Antagonists

The data analysed in this section were provided by GlaxoSmithKline. The compound set is composed of various groups. Each group is fairly homogeneous, as it is composed of congeners of compounds that were confirmed active at the early screening stage. There are also likely to be present other compounds, or compound groups, that were added later after a chemist had analysed the screening results, or incorporated domain knowledge. Some of the compounds are so similar that they are indistinguishable on the basis of physicochemical properties. For example, chiral pairs have slightly differing structures but the same molecular weight, number of hydrogen bond donors, etc. When activity is thresholded this may lead to inconsistent data. That is, a point in data space is labelled both $+1$ and -1. This is not a problem for the SVM, as one compound will be treated as an error and the other classified correctly. It does however, preclude separation of the data, necessitating the need for the soft margin formulation. (See Chapter 5, Section 5.1.1, for a discussion of this aspect of the data.)

Competitive antagonists are described in Section 3.3.2.1, together with details of the screening. The results of SVM classification of these data are presented and discussed in Section 3.3.2.2 together with some approaches that attempt to improve the accuracy of the classifier.

3.3.2.1 Competitive Antagonists

An *agonist* is a signalling molecule which binds to a receptor inducing a conformational change which produces a response. An *antagonist* is a drug which attenuates the effect of an agonist.

[2]Tom Khabaza, personal communication.

Assay	π^s_{+1}	Assay	π^s_{+1}
1	0.28	7	0.49
2	0.38	8	0.56
3	0.70	9	0.52
4	0.38	10	0.35
5	0.56	11	0.42
6	0.58		

Table 3.1: Competitive Antagonists: Proportion of active compounds (positives) in each of the training sets. Since these compounds are further downstream in the drug discovery process than those screened in HTS there is a higher proportion of actives to inactives, between 2:5 and 7:3.

Antagonists may be competitive, non-competitive or uncompetitive. A *competitive* antagonist binds to a region of the receptor which overlaps the region bound by an agonist. The agonist and antagonist compete for the same binding site and cannot simultaneously occupy the receptor. The potency of a competitive antagonist is quantified by the equilibrium dissociation constant, K_B, as determined in a functional assay. This is the concentration of antagonist which would occupy 50% of the receptors at equilibrium. This may be determined by Schild analysis (Arunlakshana and Schild, 1959) — a concentration-response curve is plotted for the agonist, in the presence of varying quantities of the antagonist. Theoretically, this value should be the same as the K_I value determined in a radioligand competition binding assay. In this assay the agonist is radio-labelled and is screened at only one concentration, usually below its equilibrium dissociation constant for the receptor, K_D. The specific level of binding of the agonist is then determined in the presence of a range of concentrations of a competing non-radioactive ligand (the antagonist). The data for each competing ligand are usually fitted to an hyperbolic equation from which the *IC50* can be determined. The *IC50* is the concentration of antagonist required to reduce the specific binding of the radioligand by 50%. This is then converted to a K_I value by the Cheng-Prusoff equation (Cheng and Prusoff, 1973),

$$K_I = \frac{IC50}{1 + \frac{[L]}{K_D}}, \tag{3.48}$$

where $[L]$ is the concentration of free radioligand used and K_D is its equilibrium dissociation constant for the receptor. Whereas the *IC50* value for a compound may vary between experiments the K_I is an absolute value. Typically, the negative logarithm of this quantity, pK_I, is reported as the measure of activity against the receptor. Since a ligand is screened over a limited range of concentrations it may not be possible to specify an exact pK_I, instead it may be reported as less than or greater than those limits implied by the concentration range.

The 1415 compounds analysed in this report were screened against 11 targets (receptors), of various classes, in competitive binding assays and the corresponding pK_I values calculated. The proportion of active compounds in each of the training sets, π^s_{+1}, is shown in Table 3.1.

The problem is to predict the pK_I values based on some set of descriptors. For the purposes of classification, a compound is treated as active if its pK_I is greater than 6. The feature set included six descriptors considered biologically relevant by GlaxoSmithKline (though not previously used to predict pK_I values), and eight one-dimensional descriptors such as molecular weight, CLOGP (Leo and Weininger, 2000), number of hydrogen bond donors and acceptors, etc. The attributes were all scaled to lie in the range $[-1, +1]$.

pK_I	C^*	Error						$AUROC$	RA
		Train	s.e	Test	s.e.	t	p		
1	1	0.16	0.02	0.22	0.01	2.66	0.00	0.77	2.77
2	10	0.10	0.02	0.27	0.01	8.28	0.00	0.76	1.65
3	1	0.14	0.02	0.26	0.01	5.40	0.00	0.72	1.05
4	1	0.13	0.02	0.25	0.01	5.53	0.00	0.78	1.82
5	10	0.14	0.02	0.33	0.01	8.32	0.00	0.72	1.29
6	1	0.17	0.02	0.37	0.02	8.26	0.00	0.68	1.12
7	10	0.04	0.01	0.32	0.01	15.89	0.00	0.73	1.36
8	10	0.02	0.01	0.27	0.01	16.04	0.00	0.78	1.37
9	1	0.14	0.02	0.31	0.01	7.49	0.00	0.74	1.38
10	10	0.04	0.01	0.23	0.01	11.55	0.00	0.80	1.86
11	1	0.11	0.02	0.24	0.01	6.29	0.00	0.82	1.83

Table 3.2: Competitive Antagonists. Error rate for SVM, with a Gaussian kernel. The test error is much higher than the training error, i.e. the SVM is overfitting the data (t and p values are shown for the null hypothesis that the SVM is not overfitting). The last two columns show the area under the ROC curve ($AUROC$) and the relative advantage (RA). C^* is that value minimizing the $\alpha\xi$ estimator.

3.3.2.2 Results and Discussion (SVC pK_i)

To evaluate the performance of the various classification algorithms the data were partitioned randomly into training and test sets of sizes 395 and 1020 respectively (this choice will be explained in Chapter 5, Section 5.1.1, below). For each of the 11 screens, an SVM was trained using the physicochemical descriptors. A Gaussian RBF kernel $K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|^2/2\sigma^2)$ was used. The RBF width σ was set using the heuristic of Jaakkola et al. (1999) as in Section 3.3.1. The parameter C was varied in $\{1, 10\}$ and chosen as that which minimized the $\alpha\xi$ estimator (3.32). (In the case of a tie, the classifier with the lower number of SVs was chosen, as motivated by (3.31).) The generalization error was estimated as the misclassification rate on the test set.

The error rates and their standard errrors on the training and test sets are shown in Table 3.2, together with the area under the ROC curve ($AUROC$) on the test data (see Section 3.2.1). The ROC curve is generated by varying the operating point b in the SVM solution, $f(\mathbf{x};\mathbf{w},b) = \text{sgn}(\langle\mathbf{w},\psi(\mathbf{x})\rangle_{\mathcal{H}} + b)$. For seven of the data sets the optimum value of C from $\{1, 10\}$ was 1, in three cases it was 10. The difference in test error between $C = 1$ and $C = 10$ was not significant for any one data set. The value of C minimizing the $\alpha\xi$ estimator was the minimizer of test error for $C \in \{1, 10\}$ in eight cases. (Under the null hypothesis that the $\alpha\xi$ chooses randomly, this has probability 0.11.) Thus, even though the classifiers obtained by setting $C = 1$ or $C = 10$ were similar, the $\alpha\xi$ estimator still tended to choose the slightly better one. It is evident from Table 3.2 that the SVM is overfitting the training data. The error rates are significantly lower on the training set than on the test set, particularly for $C = 10$. This is probably because σ is too small, see Chapter 5.

For comparison, a C5.0 decision tree, implemented in CLEMENTINE 6.0 (Clementine 6.0, 2001) was trained on the same data. The error rates and their standard errors on the test set are shown in Table 3.3. The performance of C5.0 is comparable to that of the SVM. The SVM has a significantly lower test error (t-test at 95%) on three of the eleven assays. It may be possible to determine if the SVM is performing better than C5.0 on the other assays by using the McNemar test (Ripley, 1996) or by averaging over repeated partitions of the data into training and test sets. However, the error rates are so large that the difference is of no practical importance.

A dynamic neural network (see Section 2.4.1.1) was also trained on these data. The dynamic

pK_I	C5.0				Dynamic			
	Test	s.e.	t	p	Test	s.e.	t	p
1	0.22	0.01	0.00	0.50	0.36	0.02	7.05	0.00
2	0.29	0.01	1.01	0.16	0.45	0.02	8.62	0.00
3	0.30	0.01	2.01	0.02	0.31	0.01	2.51	0.01
4	0.23	0.01	-1.06	0.15	0.36	0.02	5.43	0.00
5	0.37	0.02	1.90	0.03	0.43	0.02	4.68	0.00
6	0.38	0.02	0.47	0.32	0.43	0.02	2.77	0.00
7	0.33	0.01	0.48	0.31	0.37	0.02	2.38	0.01
8	0.32	0.01	2.48	0.01	0.33	0.01	2.96	0.00
9	0.32	0.01	0.49	0.31	0.41	0.02	4.73	0.00
10	0.25	0.01	1.06	0.15	0.30	0.01	3.59	0.00
11	0.25	0.01	0.53	0.30	0.28	0.01	2.06	0.02

Table 3.3: Competitive Antagonists. Test error rates for C5.0 and a dynamic neural network. The performance of C5.0 is comparable to that of the Gaussian SVM. The dynamic neural network performs badly. t and p values are shown for the one-tailed test that C5.0 or Dynamic has the same performance as the SVM.

neural network is implemented in CLEMENTINE 6.0, 20% of the training data are held out as a validation set to automatically optimize the network architecture. The error rates on the test set are also shown in Table 3.3. The error rates of the dynamic neural network are significantly higher than the SVM on all assays and significantly higher than those of C5.0 on eight assays.

On these data, using a Gaussian SVM does not lead to better predictive accuracy than the decision tree. This could be because the Gaussian RBF kernel is not a good choice for these data, or has not been correctly tuned. If the width parameter σ is set too small then many of the training data become SVs and the decision surface thus overfits the training data. As $\sigma \rightarrow 0$ the kernel becomes the Dirac delta function, i.e. the SVM is behaving as a memory machine. If σ is large then the decision surafce is too smooth and many SVs are again needed to capture the variation in the training data. As $\sigma \rightarrow \infty$ all of the training data are projected to the same point in feature space (since all of the inner products $\rightarrow 1$).

The number of training points that become SVs on each of these data sets is shown in Table 3.4. Also shown is the number of iterations for each run. The training requires more iterations at the higher value of C since the search space for the optimization is larger ($0 \leq \alpha_i \leq C$). This also leads to a larger function space, containing more flexible basis functions, hence fewer SVs are needed. The proportion of training points that become SVs is large (55%–85%). This is a further confirmation that the SVM is overfitting the training data.

There are a number of ways to attempt to increase the accuracy of a classifier, and to reduce the model complexity. The problem of tuning a Gaussian RBF kernel is postponed to Chapter 4 (Section 4.2). An alternative is to use domain knowledge to construct alternative representations of the objects of interest (in this case chemical compounds). This and other approaches are investigated further in Chapter 5 (Section 5.3.3).

3.3.3 Analysis of HTS Data

In this section, I return to the high throughput screening data of Section 2.4.2. The standard SVM is used to classify each of the twenty-one data sets (corresponding to the seven assays and three feature sets). The attributes were scaled to lie in the range $[-1, +1]$. In these experiments the software package LOOMS (Lee and Lin, 2000) is used. This learns an SVM classifier with a

pK_I	$C = 1$		$C = 10$	
	N_{iter}	N_{SV}	N_{iter}	N_{SV}
1	832	269	4169	238
2	1029	312	4976	264
3	1655	310	2757	282
4	1011	304	2833	273
5	936	337	3011	298
6	1537	324	2795	306
7	892	329	5256	298
8	1343	318	4307	265
9	1443	329	2473	297
10	1153	260	4150	218
11	1496	277	2495	244

Table 3.4: Competitive Antagonists. Number of iterations (N_{iter}) and number of SVs (N_{SV}) for the standard SVM. Training requires more iterations for $C = 10$ since the search space for the Lagrange multipliers is larger ($0 \leq \alpha_i \leq C$). Fewer SVs are needed to construct the decision surface at $C = 10$ as the function class is larger.

Gaussian kernel. The model parameters C and σ are chosen to minimize the leave-one-out error by performing an extensive search over hypothesis space. While such a search will introduce bias into the performance estimates (Ng, 1997) we would still expect LOOMS to have good performance. A disadvantage is that it is not possible to specify misclassification costs in LOOMS. Owing to the low proportion of positives, the SVM thus classifies all of the compounds as inactive. This is useless in practice. However, there are good reasons for expecting the unthresholded output of the SVM to be monotonic in the posterior probabitity of a compound being active (Platt, 2000). Thus, it is still possible to rank the compounds for screening set selection. Even so, the standard SVM fails to improve on the classical techniques of linear and logistic regression, despite the search over parameter space. Although much slower, the Matlab implementation (Section 3.1.3) allows one to adjust the misclassification costs. This compensates for the low hit rate and improves performance.

3.3.3.1 Ranking Compounds

Recall the performance measure of Chapter 2 (Section 2.4.2.1). A prediction method is used to rank the compounds for *in vitro* screening. The $A50$ was defined to be the number of compounds that it would be necessary to screen in order to find half of the active compounds. The global enhancement (GE) was defined to be the ratio of the $A50$ for random selection to the actual $A50$. To calculate an $A50$ it is necessary to rank the compounds according to predicted activity. For classification, a conditional probability $p(\mathbf{x}) = \Pr\{y = +1|\mathbf{x}\}$ or confidence measure is required to rank the compounds. An alternative method of ranking the compounds is to use classification to perform ordinal regression, as was done for the pyrimidine data (Chapter 2, Section 2.4.1). However, this leads to a classification data set of size $2l(l-1)$, i.e. all comparisons and their inverses, which is impractical if l is large, as it is for HTS data.

The support vector machine solution,

$$f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{H}} + b), \tag{3.49}$$

cannot be used immediately to rank the compounds. Although (3.49) is an estimator of $\text{sgn}(p(\mathbf{x}) - 1/2)$, it is not the case that $\langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{H}} + b$ is an estimator of $p(\mathbf{x})$ (Lin et al., 2000). It is possible to

	SVM			Logistic		
	fs01	fs02	fs03	fs01	fs02	fs03
a1	1.02	0.83	1.24	**1.51**	1.27	1.26
a2	2.05	2.57	**2.70**	2.48	2.57	2.07
a3	1.10	0.62	2.20	1.92	3.27	**3.49**
a5	1.91	1.35	0.95	**2.17**	1.37	1.35
a6	**18.80**	1.28	1.91	5.21	1.28	2.93
a7	1.11	0.94	1.11	**1.72**	1.08	1.66
a8	2.83	**4.54**	2.51	4.15	**4.54**	3.65

Table 3.5: HTS Results: LOOMS. The global enhancement (GE) over random screening is shown for the LOOMS software (*left*). This performs automated model order selection for an SVM with Gaussian kernel. The best performance (*emphasized*) is the same as the best classical technique (logistic regression, *right*) on one assay, poorer on four assays and better on two. The range of GE is surprisingly large. Although assay a6 with feature set fs01 was easiest for the classical techniques, the GE for the SVM is over three times higher.

obtain an estimator of $p(\mathbf{x})$ by training a kernel classifier to minimize a regularized log-likelihood score (Platt, 2000). However, this will produce a non-sparse representation of the decision function. Platt (2000) suggests fitting a sigmoid to $\langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{H}}$ after training the SVM. This requires some kind of validation to avoid over-fitting the sigmoid to the training data, and is computationally intensive. A recently suggested technique (Zhu and Hastie, 2002) is to minimize the regularized log-likelihood on only a subset of the training data. This aims to estimate $p(\mathbf{x})$ directly, whilst maintaining the sparsity of the SVM solution. A greedy forwards search strategy is required to determine a good subset of the data.

As we require only a ranking of the compounds, it is not necessary to estimate $p(\mathbf{x})$ explicitly. There is a strong reason for assuming that the unthresholded output of the SVM $\langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{H}}$ is monotonic in $p(\mathbf{x})$ (Platt, 2000). Hence we may use the unthresholded output of the SVM to rank the compounds. This approach is taken here.

3.3.3.2 Results and Discussion (SVC HTS)

The global enhancement (*GE*) of LOOMS over random screening is shown in Table 3.5. The overall performance of LOOMS is worse than that of the best classical technique, viz. logistic regression. On feature set fs01 logistic regression wins six out of seven times. On feature set fs02 logistic regression wins four out seven times, and ties with LOOMS on three. On fs03 logistic regression wins on six out of seven assays. If we assume that the best feature set could be chosen for each assay then logistic regression would win on four assays and LOOMS would win on two assays. The poor performance of LOOMS suggests that it is important to allow for the fact that the data sets are unbalanced. LOOMS performs model selection by minimizing the leave-one-out error rate. This will be minimized when predicting most or all of the compounds to be inactive. If the lowest error rate is achieved by predicting all examples to be in one class, then the lowest LOO error rate will be given by the smoothest solution. In 14 out of 21 cases, the parameters minimizing the LOO error rate were $C = 1$, the smallest considered, and $\sigma = 44$, the largest considered. The decision surface is highly oversmoothed and underfits the training data. In such a situation, the assumption that the distance of a compound in feature space to the separating hyperplane is proportional to the probability that the compound is active is unlikely to hold.

In an attempt to improve the global enhancement, an SVM was trained with the loss function (3.38). This was shown empirically, in Section 3.1.5.3, to be inversely related to the relative

	$C = 1$			$C = 1000$		
	fs01	fs02	fs03	fs01	fs02	fs03
a1	1.14	1.27	**1.40**	1.10	1.17	1.36
a2	1.80	2.56	2.36	2.59	1.25	**2.74**
a3	**4.80**	3.27	3.88	1.86	3.27	1.76
a5	1.20	1.37	1.24	**2.25**	1.37	1.30
a6	1.76	1.28	1.75	**9.12**	1.05	5.30
a7	1.52	1.08	1.54	**2.00**	0.85	1.55
a8	2.18	**4.53**	3.90	3.57	0.94	3.96

Table 3.6: HTS Results. The global enhancement (*GE*) over random screening is shown for the SVM when trained with unequal misclassification costs. The best performance (*emphasized*) is better than LOOMS on five assays, and poorer on two assays.

advantage (*RA*). The *RA* of a classifier is equivalent to the *GE* of a ranking (Section 3.2.2). These experiments were implemented in Matlab. Owing to computational time only two values of C and one value of σ were tried. As LOOMS did not provide any guidance as to a good choice of these, the following were used (Hsu and Lin, 2002). The SVM was trained with $C \in \{1, 1000\}$ and $K(\mathbf{x}, \mathbf{z}) = \exp -(\|\mathbf{x} - \mathbf{z}\|^2/d)$. When $C = 1000$ the SVM was terminated after 10000 iterations owing to excessive training time. The global enhancements are reported in Table 3.6.

This led to much denser models and non-zero true positive rates, which implies that the decision surface actually passes through the data (as opposed to the data being entirely on one side of it). Overall there was an improvement in performance. The SVM with unequal costs had a higher *GE* than LOOMS on five assays and lower *GE* on two. It had a higher *GE* than logistic regression on six assays and lower on one. This analysis assumes that the optimal feature set could be chosen for each algorithm, and the better value of C for the SVM with unequal costs could be chosen.

The risk on the test set was very close to the risk on the training set for all assays and feature sets. This implies that the SVM is not overfitting the data. It seems unlikely that the solutions could be improved by much. It may be possible to increase the *GE* by adjusting the loss functional, or by examining the ROC curve.

These data are not investigated further in this thesis. The results of this and the previous chapter are reconsidered in Chapter 6 with some suggestions for further work.

3.4 Summary

The SVM is an established learning machine for classification that has been shown to have state-of-the-art performance on a wide range of applications. The advantages of the SVM are often cited as good predictive performance, sparsity of the solution, and few free parameters. These advantages are not always obtained when the SVM is applied to classification problems from drug discovery.

The key points and contributions of this chapter are as follows.

- The few free parameters of an SVM can be set by minimizing an error or risk estimate.
  - The width, σ, of a Gaussian kernel can be set by minimizing Joachim's estimator $\varepsilon^l_{\alpha\xi}$. This is computationally efficient and provides an estimate of σ giving performance close to optimal.
  - A minor disadvantage is that it does not hold for unstable solutions. These are easily identified and, in the author's experience, are rarely optimal.

- $\varepsilon^l_{\alpha\xi}$ can also be used to estimate risk when the misclassification costs are unequal. Solutions which predict all examples to be in one class should be discarded, as $\varepsilon^l_{\alpha\xi}$ is much tighter for these solutions, and is hence misleading.
- The $\varepsilon^l_{\alpha\xi}$ estimator performed poorly for model order selection on the HTS data. This was probably because the data set has few postives.

- Care should be paid to performance analysis when the class priors are unequal. The misclassification rate may not be a useful indication of performance. The area under the ROC curve ($AUROC$) and relative advantage (RA) are more suitable measures of performance when learning decision rules for drug discovery.
  - Reweighting the misclassification costs according to the class priors can provide more meaningful solutions.
  - Minimizing this risk empirically maximizes the relative advantage, this quantity is analogous to the global enhancement.

- On a small-scale QSAR problem the SVM achieved a lower error rate than automated neural networks and a C5.0 decision tree.
  - An SVM achieved better performance (but not significantly so) than a manually tuned neural network.
  - The SVM was much faster to train than the neural networks, although it was slower than a decision tree.
  - In addition to requiring less time for model selection, tuning the SVM required much less human effort.

- In Chapter 2 it was shown that an RBF network did badly on these data, yet the SVM with equivalent kernel performed well. A sigmoid neural network performed well. These contrasts suggest the following as further work.
  - Train an RBF network with centres at the support vectors selected by the SVM. An improved RBF performance would suggest that the automated selection of centres performed by the SVM led to the improved performance.
  - Train an SVM with a kernel to mimic a sigmoid neural network with one hidden layer. An improved performance would suggest that a sigmoid is better able to describe the data than an RBF.

- In predicting the activity (pK_i) of competitive antagonists the SVM did not generally outperform a simple, quick to train decision tree, but it did outperform an automatically tuned neural network. The aim of the present work is to investigate the SVM for pharmaceutical data analysis. Whilst C5.0 may seem a better avenue of investigation based soley on this analysis, past experience of chemoinformaticians has shown that decision trees are not good enough. The vanilla SVM has not significantly outperformed C5.0. In the remainder of the thesis, I present attempts to improve the SVM.
  - Across the 11 assays $\varepsilon^l_{\alpha\xi}$ tended to select that C minimizing test error, although the differences between $C = 1$ and $C = 10$ were not large.

- The SVM overfitted the training data, this may have been due to a poor choice of the parameter σ in the kernel. The problem of automatically tuning σ is considered in the next chapter.
- The solutions had many support vectors. One of the advantages of the SVM was lost. The problem of reducing the number of SVs is considered in the next chapter.
- Predictive performance will be poor when the descriptors used do not encode sufficient information to accurately separate active from inactive compounds, contain noise, are irrelevant, or are redundant. In Chapter 5 alternative feature sets are used to learn a classifier for these data.

- A state-of-the-art automated SVM with Gaussian kernel was slightly worse than the best linear technique (logistic regression) over a range of HTS classification problems.
  - The software package LOOMS performs a search over model space, and chooses that model minimizing the LOO error. However, it has equal misclassification costs and thus predicts all compounds to be negative (since this gives the lowest error).
  - This deficiency can be partly overcome by treating the unthresholded output of the SVM as being monotone in the probability of a compound being a hit.
  - The *GE* of LOOMS was in general worse than that of logistic regression.
  - The poor performance is unlikely to be due to inadequate parameter selection, since LOOMS performs an extensive search.
  - Training an SVM to minimize a loss function related to *RA* led to higher *GE* in general than obtained by LOOMS or logistic regression.
  - With this loss functional, the solutions had many support vectors. Again, the advantage of sparsity was lost.
  - The SVM was not overfitting the data. The solutions probably cannot be improved by much.

- These data are not investigated further. Some possibilities for future work are as follows.
  - The performance attained seems near optimal for a Gaussian kernel. Future work could investigate use of alternative kernels for classifying these data.
  - To improve *GE*, *RA*, or whatever performance measure is of interest, the loss functional could be adjusted. Use could also be made of the ROC curve. (Note that *GE* is independent of the operating point, but *RA* is not.)
  - Logistic regression performed well. Incorporating the flavour of the SVM would suggest regularized kernel logistic regression.

The SVM has been introduced and its advantages and disadvantages discussed and demonstrated on public domain and drug discovery data sets. In the next chapter, heuristics are introduced to overcome some of the disadvantages. In Chapter 5, these are applied to the analysis of the pK_i data sets.

Chapter 4

Heuristics for SVC

> ***heuristic*** ... Computing *proceeding to a solution by trial and error*
>
> THE CONCISE OXFORD

In the previous chapter, the SVM for classification was introduced. Its advantages and disadvantages were discussed and demonstrated on some public domain and drug discovery data sets. The disadvantages highlighted in the analysis of the GlaxoSmithKline data motivated the development of the heuristics presented in this chapter. In Section 4.1, a new stopping criterion for the SVM is introduced. This allows early termination of the SVM without loss in predictive performance. This stopping criterion is also useful when using the other heuristics, since the modified SVM is no longer guaranteed to converge to a global (or even local) optimum. In Section 4.2, the LAIKA heuristic for tuning the width of a Gaussian kernel is introduced and developed. In Section 4.3, the STAR heuristic for reducing the number of support vectors is described. The results and conclusions are summarized in Section 4.4.

4.1 Stopping Criteria

[1] In many practical data-mining situations it is not necessarily desired to obtain a global maximum, assuming one exists. Many existing applications of machine learning to data-mining have many local optima, e.g. neural networks (Hertz et al., 1991) and decision trees (Breiman et al., 1984). The SVM, in contrast, is the solution to a convex quadratic program and as such has a global optimum.

Work on reducing the run-time complexity of support vector classification has thus far concentrated on decomposing the quadratic program (Joachims, 1999a; Platt, 1999; Hsu and Lin, 2002) or developing on-line perceptron like algorithms (Freund and Schapire, 1999; Frieß et al., 1998; Gentile, 2001; Li and Long, 2001). The stopping criteria of these algorithms are defined in terms of the optimization problem. From a data-mining point of view this may not be necessary. For example, during exploratory data analysis the data miner usually desires to obtain approximate solutions for a variety of algorithms before deciding which is most suited to the task in hand. For this purpose one can specify various stopping criteria. In designing an SVM toolkit therefore one should include stopping criteria based on training time, number of cycles through the data set (for online algorithms), and expected generalization error.

[1] This work has recently been accepted for publication (Burbidge, 2002b).

In the following section, a heuristic stopping criterion for support vector classification is presented. In Section 4.1.2, the performance of the SVM with this stopping heuristic is compared to the performance of the classifier obtained by training until a global optimum is found. Stopping the SVM early can lead to a substantial reduction in training time, with little or no loss in accuracy. This is especially the case when the regularization constant C is high.

4.1.1 Persistence

The expected generalization error is of particular relevance to support vector classification of complex data sets. For example, for the Diabetes data set analysed in Chapter 3 (Section 3.1.5.3) convergence to the global optimum of the QP can be very slow for certain kernels and parameter settings. The same phenomenon is observed on many data sets provided by GlaxoSmithKline. However, on all of these data sets solutions sub-optimal in terms of the QP perform as well as the global optimum in terms of classification risk. This may be related to observations about the hypothesis space made by Herbrich et al. (2002) namely that there are a large number of hypotheses that have the same performance characteristics. These sub-optimal solutions to the QP can often be found in considerably less training time than the global optimum of the QP.

In order to assess the quality of a solution, optimal or otherwise, use can be made of error bounds or estimators. The $\alpha\xi$ estimator (Joachims, 2000, see also Chapter 3, Section 3.1.5.2) can be evaluated at little cost during training. This quantity is an estimator of the expected leave-one-out (LOO) error over the data set, which is an almost unbiased estimator of the expected generalization error. However, this estimator is only valid at the global optimum. Theoretically, this is when $\epsilon = 0$ in (3.25). In practice, the SVM is only trained to $\epsilon = 0.001$. Lee and Lin (2000) have shown, on a number of publicly available data sets, that the LOO error rate of an SVM trained to a precision of $\epsilon = 0.1$ is qualitatively the same as that of an SVM trained to a greater precision. This result has also been confirmed on the competitive antagonist data analysed in Chapter 3 (Section 3.3.2) and Chapter 5 (Section 5.3.3).[2] This empirical evidence suggests that the $\alpha\xi$ estimator may be used heuristically during training to a give a rough measure of the relative change in estimated risk.

Thus one can specify a value for 'persistence' and terminate the algorithm when this number of iterations (or cycles through the data set for online algorithms) have been performed with little or no change in the value of the $\alpha\xi$ estimator. Such criteria are often used for termination of neural network training, e.g. as in Clementine (SPSS, 1999). The advantage of using such a criterion for an SVM however is that little computation is required to evaluate it. When using such a stopping criterion for a neural network a subset of the training data, known as a hold-out or validation set, must be witheld from training. The neural network is evaluated on this set to provide an estimate of the true error. This method has two disadvantages: not all of the available data are used for training, and evaluating the learned decision function on the hold-out set introduces further computation. Both of these disadvantages are avoided with an SVM since the $\alpha\xi$ estimator is calculated at little cost from the α and ξ values, a hold-out set not being required.

The behaviour of the $\alpha\xi$ estimator and of the test error of an SVM during training is illustrated in Figure 4.1 for the Diabetes data set (see Chapter 3, Section 3.1.5.3 above) and the following data set.

Ionosphere This data set was created by Vince Sigillito at Johns Hopkins University, Maryland (Sigillito et al., 1989). There are 350 examples of radar returns, characterized by 34 attributes.

[2]Steven Barrett, GlaxoSmithKline, personal communication.

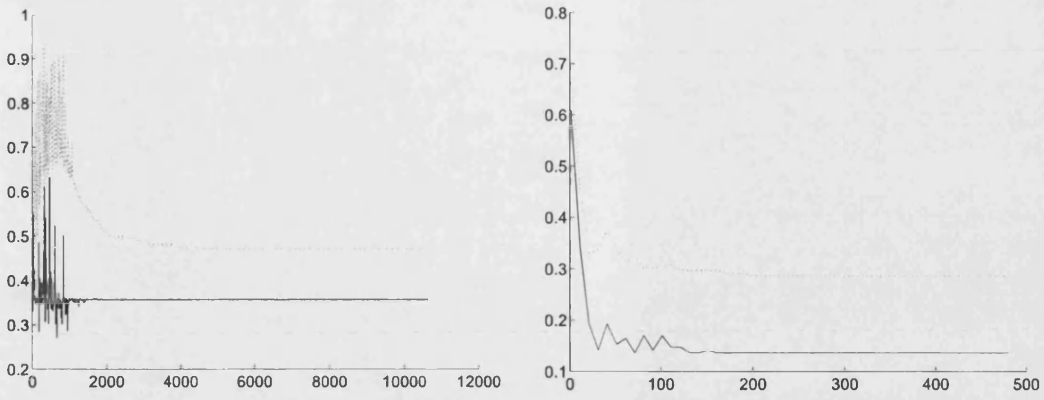

Figure 4.1: HERMES. An SVM with Gaussian kernel was trained on 50% of the data, with $C = 1000$. The $\alpha\xi$ estimator (*dotted line*) is qualitatively the same as the test error (*solid line*) during the later stages of training. The behaviour of the $\alpha\xi$ estimator can thus be used to define a heuristic stopping criterion. Note that the test error attains its minimum after only a few iterations. Moreover, the test error is rather poor, both for the Diabetes data (*left*) and the Ionosphere data (*right*), thus stopping early is advisable when searching for a good value of C or σ.

> The problem is to discriminate between 'good' radar returns ($y = +1$) from the ionosphere and 'bad' radar returns ($y = -1$). The proportion of good returns is $\pi^s_{+1} = 0.64$.

A Gaussian kernel was used with $\sigma = \sqrt{d/2}$, where d is the number of input attributes, and $C = 1000$. The data were split into training and test sets of equal sizes. The figure illustrates that the test error reaches its minimum many iterations before the termination criteria for the QP are satisfied. The $\alpha\xi$ estimator also levels off when the minimum of the test error is attained, suggesting that, at least under these conditions, the above heuristic is a useful stopping criterion. It should be noted that, unlike a neural network trained with an iterative gradient-based method — where stopped minimization can prevent overfitting (McLoone and Irwin, 2001) — the SVM does not begin to overfit the training data with increased training.

A further point is that the test error on both data sets is rather high, which implies that the choice of C and σ may be poor. It is desirable to stop training early during parameter selection to avoid wasting effort on such, possibly poor, parameter settings. In the following, this heuristic will be termed HERMES (Heuristic Error Rate Method for Early Stopping).

4.1.2 Results and Discussion (HERMES)

The data sets used in the evaluation of HERMES were the Diabetes data (see Chapter 3, Section 3.1.5.3) and the following.

Heart This data set was created by Robert Detrano at the V.A. Medical Center, Cleveland. (The version used here is that used in Statlog (Michie et al., 1994)). There are 270 examples of patients characterized by 13 physiological attributes. The task is to discriminate between absence ($y = -1$) and presence ($y = +1$) of heart disease. The proportion of patients with heart disease is $\pi^s_{+1} = 0.44$.

Adult This data set was created by Ron Kohavi and Barry Becker at Silicon Graphics (Kohavi, 1996). The two data sets used here, Adult1 (1605 examples) and Adult4 (4781 examples), are the versions used by Platt (1999). The data are census data. The problem is to predict whether or not a person earns more than \$50K per year ($y = +1$), based on 14 attributes, such

C	1	5	10	1000
Australian	0.1449	0.1377	0.1420	0.1783
Diabetes	0.2250	0.2250	0.2289	0.2724
Heart	0.1741	0.2000	0.2000	0.2407
Adult1	0.1700	0.1694	0.1675	0.2131
Adult4	0.1626	0.1634	0.1651	0.2249

Table 4.1: HERMES. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$, and persistence of 100. Individually, the results are not significantly different, at 95%, from the standard SVM.

C	1	5	10	1000
Australian	0.1464	0.1377	0.1435	0.1783
Diabetes	0.2307	0.2281	0.2307	0.2568
Heart	0.1778	0.1926	0.2000	0.2482
Adult1	0.1682	0.1701	0.1738	0.2118
Adult4	0.1625	0.1671	0.1728	0.2248

Table 4.2: BSVM. 10 fold CV error rates (Hsu and Lin, 2002). The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$. Individually, the results are not significantly different, at 95%, from using HERMES.

as marital status, occupation, etc. There are eight categorical variables and six continuous variables. The continuous variables have been discretized into quintiles. This results in 123 binary variables in total. The proportion of persons earning over \$50K per year is $\pi^s_{+1} = 0.25$.

Australian This data set consists of 690 examples of credit card applications, described by 14 attributes (Quinlan, 1987). The problem is to predict whether an application is approved ($y = +1$) or declined ($y = -1$). Missing values have been replaced by the mean (for continuous attributes) or the mode (for nominal attributes). Nominal attributes are coded as integers. The proportion of applications that were approved is $\pi^s_{+1} = 0.68$.

All of the attributes were scaled to lie in the range $[-1, +1]$. The SVM was trained with a Gaussian kernel $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|/d)$, where d is the dimensionality of the data. The SVM was trained for $C \in \{1, 5, 10, 1000\}$. The persistence was set at 100, i.e. the SVM was halted if there had been no change in $\varepsilon^l_{\alpha\xi}$ for 100 iterations.[3]

The results for the standard SVM are those reported by Hsu and Lin (2002) for their software BSVM. The Matlab implementation used for HERMES uses the same QP formulation and the same chunking strategy as BSVM (and the same parameters). There are minor differences in the number of iterations of the Matlab implementation and BSVM since the Matlab implementation solves the subproblem to machine precision using the Hildreth and D'Espo algorithm, whereas BSVM uses TRON (Lin and Moré, 1999). The Matlab implementation requires about 10% fewer iterations on the Heart and Adult data sets. (In terms of absolute runtime BSVM is much quicker since it is implemented in C. Hence, the number of iterations is recorded here for purposes of comparison.)

The 10 fold cross-validated error rates of the SVM with HERMES are shown in Table 4.1. These are almost identical to those obtained by training the SVM to a global optimum, see Table 4.2. The only notable differences are for Adult4, with $C = 10$, where early stopping did slightly better ($p = 0.08$), and Diabetes, with $C = 1000$, where BSVM did slightly better ($p = 0.16$).

[3]Recall that $\varepsilon^l_{\alpha\xi}$ is a discrete variable.

		HERMES			BSVM		
Data	l	N_{SV}	N_{err}	N_{iter}	N_{SV}	N_{err}	N_{iter}
Australian	690	245	98	205	245	98	200
Diabetes	768	447	168	132	447	168	131
Heart	270	130	36	47	130	36	53
Adult1	1605	695	243	261	691	228	293
Adult4	4781	1877	716	765	1888	700	841

Table 4.3: HERMES. $C = 1$. Since the number of iterations required is fairly small when C is small, the SVM converged before the HERMES criterion was satisfied. The minor differences in the results are due to the differing implementations of the QP solver.

		HERMES			BSVM		
Data	l	N_{SV}	N_{err}	N_{iter}	N_{SV}	N_{err}	N_{iter}
Australian	690	242	28	4304	222	28	12609
Diabetes	768	403	127	4939	377	128	21908
Heart	270	102	0	706	101	0	1230
Adult1	1605	690	16	3177	649	14	9098
Adult4	4781	1987	110	27731	1811	97	85103

Table 4.4: HERMES. $C = 1000$. The SVM was halted by the HERMES criterion on all five data sets. The reduction in the number of iterations (N_{iter}) required varies from 43%–77%. The price paid is a slight increase in the number of support vectors (N_{SV}) and the number of training errors (N_{err}).

The number of iterations required by the SVM with and without early stopping are shown in Table 4.3 and Table 4.4 for $C = 1, 1000$ respectively. Also shown are the number of support vectors and the number of training errors. For the smaller value of C the SVM converged before the HERMES criterion was satisfied. For the larger value of C the SVM terminated on the HERMES criterion on all five data sets. The reduction in the number of iterations varied from 43%–77%, although up to 10% of this is due to the different QP solvers used. The price paid is a slight increase in the number of support vectors (1%–10%) and in the number of training errors (0%–14%). These increases are related since training errors become SVs. These could be removed heuristically by the RaR method of Burbidge et al. (2001b) or by STAR (see Section 4.3, below).

Training past a certain point, then, does not improve the generalization accuracy of the SVM, but it can slightly reduce the number of support vectors and the number of training errors. The stopping criterion HERMES can substantially reduce the training time of an SVM, without a reduction in generalization accuracy. The most significant increase ($p = 0.16$) in generalization error when stopping early was on the Diabetes data, with $C = 1000$. This also corresponded to the greatest reduction in the number of iterations. This suggests that, as with all heuristics, one should take care to validate the solution obtained. Finally, note that BSVM was specifically developed for the case where C is high and was shown to require substantially fewer iterations at $C = 1000$ than the then state of the art SVM^{light} (Hsu and Lin, 2002; Joachims, 1999a).

4.2 Automated Model Order Selection

The simplest kernel for data in a Euclidean space is the linear kernel $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$, which uses the scalar product between examples as the similarity measure. Other popular kernel functions are designed to mimic well-known classifier systems, such as the polynomial kernel $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z}+1)^d$ and the Gaussian radial basis function kernel (RBF) $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/2\sigma^2)$. When using

a Gaussian RBF kernel the scale parameter σ must be specified in advance. This parameter can be thought of as the width of the RBF. It controls the trade-off between faithfulness to the training data and smoothness of the decision surface in the input space. When σ is too small the SVM overfits the training data, essentially every training point becomes an RBF centre (support vector). If σ is too large then the SVM underfits the training data, as it is not able to adequately capture the inherent complexity of the data generating process. The quality of the solution depends on the choice of σ (see Figures 3.2–3.7).

One basic approach is to train the SVM for a range of values of σ and use that value which minimizes some estimate of the generalization error (e.g. those given in Chapter 3, Sections 3.1.5.1 and 3.1.5.2). This approach is prohibitive in terms of training time, which for an SVM has been found empirically to scale quadratically in the sample size (Platt, 1999; Joachims, 1999a).

In Section 4.2.1, below I describe some incremental approaches to automatically tuning kernel parameters. These approaches train the SVM for a range of parameters and choose that parameter minimizing an error bound. In the subsequent sections, I present and refine a heuristic for updating the width parameter σ during optimization. This heuristic is evaluated on a number of publicly available data sets and compared to the underlying (fixed) heuristic on which it is based.

4.2.1 Incremental Tuning

Consider the case of a Gaussian kernel with one free parameter σ. The most basic approach to model order selection is to train an SVM for a range of values of σ and choose that value minimizing some estimate of the generalization error. The generalization error, ε, of an hyperplane classifier with margin γ can be upper bounded as follows (Vapnik, 1995):

$$\varepsilon \leq O\left(\frac{R^2}{l\gamma^2}\right), \tag{4.1}$$

where R is the radius of the smallest ball containing the training set and l is the number of training points. The radius R is approximable by 1 for a Gaussian kernel (see Chapter 3, Section 3.1.5.1). Cristianini et al. (1998) show that this bound is smooth in σ. They suggest the following procedure for Gaussian kernels.

1. Initialize σ to a very small value.
2. Maximize the margin.
   - Compute the error bound.
   - Increase the kernel parameter $\sigma \leftarrow \sigma + \delta\sigma$.
3. Stop when a predetermined value of σ is reached, else repeat 2.

The motivation here is that for small σ convergence is rapid, and few iterations will be required to bring the solution back to the maximal margin solution after each update. This approach, when implemented with the kernel-adatron algorithm (Frieß et al., 1998), provides a reasonable value of σ for good generalization. The experiments reported used hard margin SVMs, for which there is no regularization parameter C.

When there is more than one parameter to choose for the SVM the above approach becomes computationally infeasible. For example, Lee and Lin (2000) consider 14 values of σ and 11 values of C for soft margin SVMs with Gaussian kernels. This leads to 154 parameter sets. To evaluate the performance of the SVM on all 154 parameter sets in a reasonable amount of time the authors

use some numerical tricks to calculate efficiently the actual leave-one-out error rate. Further, in calculating the error rate the stopping precision used in (3.25) is $\epsilon = 0.1$, instead of the more precise and more standard $\epsilon = 0.001$. This relaxed criterion allows an approximate solution and error rate to be calculated more rapidly and provides close to optimal parameter selection over the ranges tested.

A more principled approach to parameter selection is provided by Chapelle et al. (2002). If the errors ξ_i in (3.15) are quadratically penalized, corresponding to $k = 2$, then the regularization parameter C can be treated as a kernel parameter. The kernel matrix in the Wolfe dual is modified according to $K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + 1/C$ and the constraint $C \geq \alpha_i \geq 0$ in (3.18) becomes $\alpha_i \geq 0$. Within this framework it is possible to calculate the differential of an error bound with respect to the kernel parameters. Chapelle et al. (2002) calculate the derivative of the VC bound (4.1) and of the span bound. The span bound (Chapelle and Vapnik, 2000) is a tight upper bound on the leave-one-out error rate, but is not continuous in the kernel parameters. In order to overcome this a regularized version is used. This introduces a further parameter.

Starting from small values of C and σ for a Gaussian kernel the SVM is trained and the gradient of the bound calculated. A step is taken in parameter space in the direction of maximum decrease in the error bound and the SVM is retrained. They compared this approach to estimating error rate by five-fold cross-validation for 10 values each of C and σ, as described in Rätsch et al. (2001). The main advantage of the gradient descent approach is that the number of SVMs trained is much less than the 500 required by the cross-validation approach. In practice, Chapelle and Vapnik (2000) recommend performing the gradient descent with respect to the VC bound (4.1) as this is easier to calculate than the span bound and has fewer local minima. Note that the VC bound (4.1) can only be used for non-separable data when the errors are penalized quadratically.

4.2.2 Heuristic Approaches

The above approaches to adaptive tuning of SVMs suffer from two disadvantages. Firstly, they all require sequential retraining of the SVM for a range of values of σ. The solution to each successive quadratic program should be close to the previous one since the margin depends smoothly on σ (Cristianini et al., 1998). This results in an approximate doubling of the training time for the method of Cristianini et al. (1998). Chapelle and Vapnik (2000) also report an approximate doubling of training time. Secondly, the gradient descent algorithms have an additional layer of machinery. This introduces extra parameters regarding the training rate and extra computation in calculating the gradients.

One aspect common to all of the above techniques is that σ is tuned according to *predicted error*. An alternative approach is to tune σ to be on the scale of the margin of separation, i.e. to let the data dictate directly the value of σ. An RBF network is usually trained by first using some clustering technique to choose the RBF centres $\mathbf{c}_i$ and widths σ_i and then performing a classification (or regression) step in the feature space implicitly defined by the RBFs (Bishop, 1995; Ripley, 1996). Bishop (1995) suggests setting all σ_i equal to some multiple of the average separation of the centres. To allow for different widths in different areas of space, the widths may be set to be the average distance from each basis function to its k-nearest neighbours, for some small value k. Jaakkola et al. (1999) suggest a similar heuristic for the RBF kernel of an SVM,

$$\sigma_{\text{JDH}} = \underset{i|y_i=-1}{\text{median}} \left(\min_{j|y_j=+1} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right), \tag{4.2}$$

That is, σ is set to be the median separation of each example to its nearest neighbour in the other class. The positive and negative examples are not transposable in this definition. If the classes are highly imbalanced, e.g. $\pi_{+1} < 0.01$, then the median should be taken with respect to the under-represented class. The difference is negligible for the data sets considered here.

Since the SVM solution depends only on the support vectors, which are the RBF centres, it seems desirable to apply this heuristic to the support vectors only. The simplest way to do this would be to train the SVM, calculate σ from the set of support vectors obtained, and retrain with the updated σ. This process could be repeated until the solution appeared to stabilize (although there is no guarantee that this would happen). This simple procedure involves retraining the SVM for a range of values of σ and is hence undesirable. In the following sections, I describe and analyse some online approximations to this process.

4.2.3 Approximating the Support Vector Set

Since we wish to apply the above heuristic only to the basis function centres (support vectors) an approximation to the set of support vectors is required. LAIKA (locally adaptive[4] iterative kernel approximation) updates σ during training based on some estimate of the final support vector set. Initially, the width is set to be σ_{JDH}. After every h_a iterations of the decompostion algorithm σ is updated according to:

$$\sigma = \underset{i \in A}{\text{median}} \left(\min_{j \in B} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right), \tag{4.3}$$

where A and B are as described below.

Version 1.0 of LAIKA (Burbidge, 2002a) updates σ on the *current* SV set, that is at the t^{th} iteration (where t is some multiple of h_a),

$$\begin{aligned} A(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ y_i = -1\}, \\ B(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ y_i = 1\}, \end{aligned}$$

where $\alpha_i(t)$ is the Lagrange multiplier of $\mathbf{x}_i$ at the t^{th} iteration. Initially, the sets A and B will provide very poor approximations to the sets of SVs. Nearer convergence, the approximations will improve.

Burbidge (2002a) demonstrated that this algorithm converged to a value of σ close to that predicted by a line search to minimize the $\alpha\xi$ estimator, for the Cancer, Heart and Ionosphere data sets. On the Diabetes data set, the 10 fold CV error of LAIKA was significantly worse than that obtained by the line search. It was postulated that this may be due to the high number of training errors and support vectors. The main advantage of using LAIKA instead of a line search was that the training time was reduced by a factor of four to 200 for the various data sets.

This algorithm was also implemented on the pharmaceutical classification problem of Section 3.3.2 and was shown to suffer from some disadvantages. The results are discussed more fully in Chapter 5 (Section 5.2.2). The data sets consisted of overlapping classes, with the SVM error rate being around 25%. This led to dense solutions with the majority of the training data becoming support vectors. That is, the median separation of SVs was approximately the same as for the entire data set. The adaptive tuning had little effect. Also, the 25% of the training data that become errors are also SVs — thus, errors contribute to the adaptive tuning, which is intuitively unsound.

[4]Note that *adaptive* is used here in the standard English sense, not in the statistical sense meaning convergence to the optimal value.

Version	$h_a = 20$	40	60	80	100
1.0	0.0360	0.0360	0.0383	0.0394	0.0377
1.1	0.0323	0.0323	0.0323	0.0323	0.0326
1.2	0.0329	0.0331	0.0331	0.0334	0.0329

Table 4.5: LAIKA Version 1: Cancer, $C = 1$. The test error is almost constant for different values of the update frequency h_a. Versions 1.1 and 1.2 slightly outperform version 1.0. The errors are lower than for $C = 10$.

Version	$h_a = 20$	40	60	80	100
1.0	0.0526	0.0531	0.0531	0.0526	0.0531
1.1	0.0469	0.0471	0.0471	0.0474	0.0477
1.2	0.0460	0.0469	0.0469	0.0463	0.0463

Table 4.6: LAIKA Version 1: Cancer, $C = 10$. The test error is almost constant for different values of the update frequency h_a. Versions 1.1 and 1.2 slightly outperform version 1.0. The errors are higher than for $C = 1$.

Version 1.1 of LAIKA attempts to address these problems by using

$$\begin{aligned} A(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ \xi < 1 \ \& \ y_i = -1\}, \\ B(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ \xi < 1 \ \& \ y_i = 1\}, \end{aligned}$$

that is, letting σ depend only on the current set of *correctly classified* SVs. This should overcome some of the disadvantages of version 1.0. Also, since fewer points are being used in the calculation, it is quicker to evaluate. An extension of this is version 1.2 for which

$$\begin{aligned} A(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ \xi = 0 \ \& \ y_i = -1\}, \\ B(t) &= \{\mathbf{x}_i | \alpha_i(t) > 0 \ \& \ \xi = 0 \ \& \ y_i = 1\}, \end{aligned}$$

that is, σ depends only on support vectors that lie on the edge of the margin band (i.e. are not margin errors). For separable data, these constitute all of the SVs, whereas for non-separable data these are termed *free* SVs.

Since the objective is to automatically tune one of the few free parameters of the SVM the question arises as to whether this has been achieved. The above heuristics update σ according to (4.3) every h_a iterations. This appears to have removed the parameter σ at the expense of introducing a new parameter h_a. The parameter h_a is not strictly required, as the most obvious thing to do is to update on every iteration. After each update it is required to recalculate the cached values of the gradient and the Hessian for the quadratic program. To speed up training time it is advisable to update σ only periodically.

The above three versions of LAIKA were used to classify the Cancer data set (see Section 3.1.5.3). The update frequency h_a was varied in $\{20, 40, 60, 80, 100\}$, for $C \in \{1, 10\}$. The algorithm was trained on 50% of the data and evaluated on the remaining 50%, all results are averaged over ten such runs. Since the algorithm is not guaranteed to converge, the maximum number of subproblem iterations was set at 10000. The test errors are shown in Table 4.5 for $C = 1$ and Table 4.6 for $C = 10$. For this data set the update frequency has almost no effect on the test error, for either value of C. This data set is not particularly challenging in my experience — there is a large region in parameter space that gives the same error rate.

A more challenging data set is Diabetes (see Section 3.1.5.3). The optimal region in parameter

Version	$h_a = 20$	40	60	80	100
1.1	0.2516	0.2520	0.2513	0.2516	0.2516
1.2	0.2402	0.2422	0.2406	0.2406	0.2399

Table 4.7: LAIKA Version 1: Diabetes, $C = 1$. The test error is almost constant for different values of the update frequency h_a. Versions 1.2 slightly outperforms version 1.1. The errors are lower than for $C = 10$.

Version	$h_a = 20$	40	60	80	100
1.1	0.2832	0.2832	0.2835	0.2832	0.2832
1.2	0.2721	0.2728	0.2725	0.2721	0.2721

Table 4.8: LAIKA Version 1: Diabetes, $C = 10$. The test error is almost constant for different values of the update frequency h_a. Versions 1.2 slightly outperforms version 1.1. The errors are higher than for $C = 1$.

space is smaller, see Lee and Lin (2000) for visualizations of the error surface for this and other publicly available data sets. The test errors are shown in Table 4.7 for $C = 1$ and Table 4.8 for $C = 10$. Owing to excessive training time and poor performance, version 1.0 was not fully evaluated on this data set. Again, the update frequency has almost no effect on the error rate. Version 1.2 slightly outperforms version 1.1.

In Section 3.1.5.3, an SVM was trained on both of these data sets (with $C = 1$) for a range of values of the width parameter σ. The error rate achieved by LAIKA *in one run* is the minimal error rate achievable on these data, under these conditions.

4.2.4 Reducing Training Time

The objective of this approach has been to adaptively tune the SVM without increasing training time. In an attempt to reduce training time the heuristics were adapted. Instead, of updating σ according to the present set of (correct, free) support vectors, σ is updated only on those points that have been (correct, free) support vectors for the last h_a iterations. Analogously to shrinking (Joachims, 1999a) and STAR (Burbidge et al., 2001b), these points are more likely to be (correct, free) support vectors in the final solution. Updating only on these points requires less computation, and should avoid unnecessary deviations during learning caused by adapting σ to points which will not appear in the final solution. The corresponding heuristics are termed versions 2.0, 2.1, and 2.2. For these three versions the update frequency had almost no effect on the error rates, which were the same as versions 1.0, 1.1 and 1.2, respectively, on the Cancer data. On the Diabetes data the error rates for versions 2.1 and 2.2 were the same as those for versions 1.1 and 1.2 respectively. In general, the test accuracy only depended upon the type of support vector used in the calculation of σ and not upon the update frequency. Updating on free SVs gave the best results, followed by updating on correct SVs and updating on all SVs.

Since the update rate h_a does not affect the error rate it may be possible to select it such that the training time is lowest. The effect of h_a on training time is shown in Figure 4.2 for the Cancer data set and Figure 4.3 for the Diabetes data. Figures are only shown for $C = 1$ as this was optimal for both data sets. The graphs for $C = 10$ are similar, with training times being of the same order.

These figures show that a high value of h_a leads to shorter training times. This is to be expected since there are fewer updates and hence less computation. Increasing h_a further may lead to shorter training times but ultimately the behaviour of LAIKA will be lost. Over all of the experiments

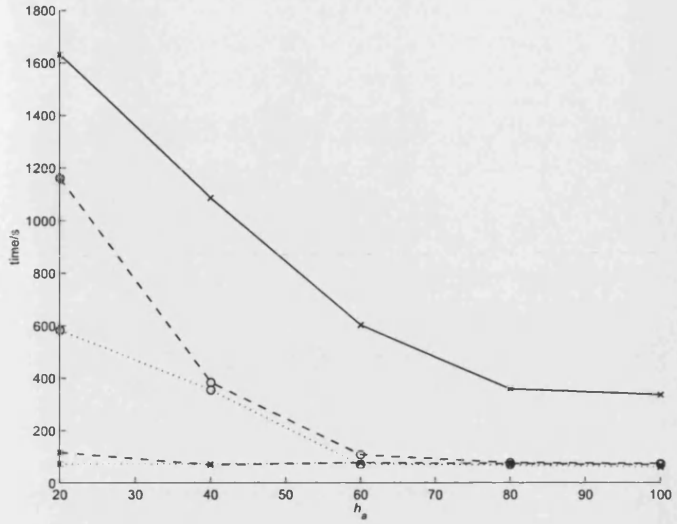

Figure 4.2: LAIKA: Cancer $C = 1$. Training time decreases as h_a increases, since fewer updates are being made. Versions 2.1 (*crossed, dashed line*) and 2.2 (*crossed, dotted line*) are fastest. Versions 1.1 (*circled, dashed line* and 1.2 (*circled, dotted line*) approach the fastest as h_a increases. Version 2.0 (*crossed, solid line*) is slowest.

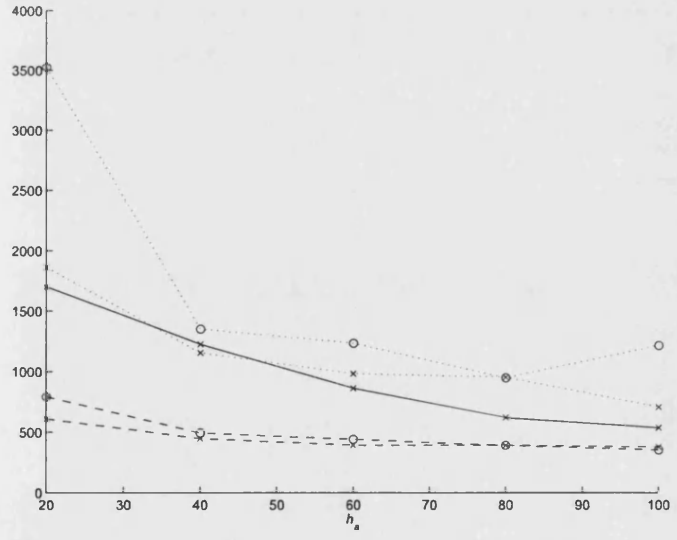

Figure 4.3: LAIKA: Diabetes $C = 1$. Training time decreases as h_a increases, since fewer updates are being made. Versions 1.1 (*circled, dashed line*) and 2.1 (*crossed, dashed line*) are fastest. Versions 1.2 (*circled, dotted line*), 2.0 (*crossed, solid line*) and 2.2 (*crossed, dotted line*) approach the fastest as h_a increases.

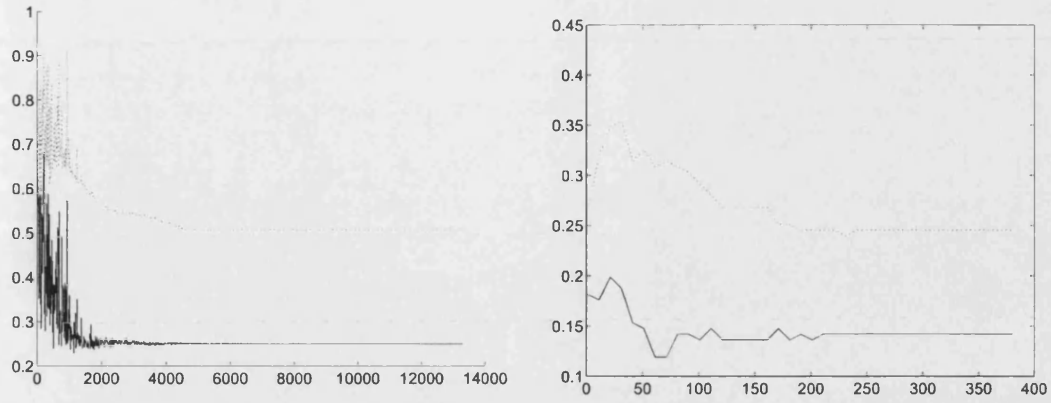

Figure 4.4: HERMES and LAIKA. As in Figure 4.1, the $\alpha\xi$ estimator (*dotted line*) is qualitatively the same as the test error (*solid line*) during training. Note that the test error is much lower on Diabetes when using LAIKA than using the original value (cf. Figure 4.1).

version 2.1 tended to be fastest. Version 2.2 tended to give the lowest error rates but sometimes failed to converge within 10000 iterations. Version 2.2 is actually the quickest for those cases where it does converge. If an alternative stopping criterion were available (other than optimality of the QP), then version 2.2 would be the favoured heuristic. Such a criterion was proposed in Section 4.1. A further motivation for preferring version 2.2 is that it usually gives the sparsest solution (i.e. the one with the fewest support vectors), and always gives a solution sparser than that of version 2.1. This seems to confirm the motivation behind LAIKA. The RBF centres should ideally be free support vectors, since these are less likely to be noisy points, and the RBF width should be on the scale of their separation, such that fewer SVs are needed to construct the boundary.

4.2.5 Stopping Criterion

It is proposed to use version 2.2 of LAIKA to automatically tune the parameter σ during training. However, as noted in the previous section, this heuristic is not guaranteed to converge and an alternative stopping criterion is required. Such a heuristic criterion was presented in Section 4.1.1 above. One should be wary of combining heuristics however.

The behaviour of the $\alpha\xi$ estimator and the test error of an SVM during training with LAIKA is illustrated in Figure 4.4 for the Diabetes and Ionosphere data sets (see Chapter 3, Section 3.1.5.3 and Section 4.1.1, above, respectively, for details). The update frequency was set at $h_a = 20$. Again, $C = 1000$ and the data were split into training and test sets of equal sizes. The same conclusions can be drawn as before, suggesting that HERMES and LAIKA are compatible.

4.2.6 Results and Discussion (LAIKA)

The chief objective of the above work has been to develop a heuristic for obtaining a Gaussian SVM solution without prespecifying the width parameter σ. The development of the heuristics was motivated by the observations of Burbidge (2002a) regarding the heuristic of Jaakkola et al. (1999) and LAIKA version 1.0. On complex or noisy data sets, where $\sim 20\%$ of the training data become training errors, these heuristics underestimate σ. Given this motivation, it is appropriate to evaluate the performance of the refined heuristic on such data sets. The data sets: Diabetes, Heart, Adult1, Adult4 and Australian are used here. These vary in the number of examples and

C	1	5	10
Australian	**0.1449**	0.1580	0.1551
Diabetes	**0.2474**	0.2592	0.2803
Heart	**0.1889**	0.2037	0.1926
Adult1	**0.1712**	0.1775	0.1837
Adult4	**0.1649**	0.1766	0.1866

Table 4.9: SVM. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with width σ_{JDH}. That minimum error over the three values of C is emphasized.

C	1	5	10
Australian	0.1449	0.1478	**0.1420**
Diabetes	**0.2276**	0.2526	0.2513
Heart	0.1926	0.1889	**0.1815**
Adult1	0.1681	**0.1662**	0.1725
Adult4	**0.1703**	0.1707	0.1887

Table 4.10: LAIKA. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with the width parameter updated by LAIKA version 2.2. That minimum error over the three values of C is emphasized.

dimensions, and in the types of the attributes and preprocessing, but all typically have 15%–25% training errors after SVM classification.

For the SVM, σ was set using the original heuristic σ_{JDH} (4.2). This value was also used as the initial value (σ_0) for LAIKA. LAIKA version 2.2 is used here, as this was the preferred version. The update period was $h_a = 100$ iterations. HERMES is used as the stopping criterion with a persistence of 100.

The 10 fold cross-validated (CV) error rates of the SVM are shown in Table 4.9. The error rates for LAIKA are shown in Table 4.10. The error rate of LAIKA is lower than that of the SVM except on Heart with $C = 1$ and Adult4 with $C = 1, 10$. None of the individual differences are significant. The p-values for the two-tailed t-test of the null hypothesis that LAIKA and the SVM have the same error rate over the five data sets are 0.1292, 0.0540, and 0.0234 for $C = 1, 5, 10$, respectively.

If $\varepsilon^l_{\alpha\xi}$ had been used to select the optimal value of $C \in \{1, 5, 10\}$ then LAIKA would have won three times and the standard SVM once (with one draw). This is because $\varepsilon^l_{\alpha\xi}$ selected the optimal value of C on only two data sets when using LAIKA. For the standard SVM, $\varepsilon^l_{\alpha\xi}$ selected the optimal value of C on all five data sets. One should be wary of combining too many heuristics.

The optimal value of C for the SVM was 1 for all five data sets. The performance deteriorated as C was increased. For the higher values of C, LAIKA partially compensated for these poorer choices of C (this is reflected in the p-values quoted above). The coefficient of variation of error with C was on average about 30% less than that of the SVM. Overall, any improvement in accuracy in using LAIKA over using the original heuristic is minor and not significant for any individual problem.

The training times of the SVM and the SVM with LAIKA are shown in Table 4.11 and Table 4.12 respectively. Training with LAIKA took on average 35% longer than training the SVM. The training time of LAIKA varied haphazardly with the size of the data set and the value of C. On average, the number of iterations of LAIKA was the same as for the SVM. The number of iterations was in general reduced for the larger data sets and larger values of C and increased in the opposite scenario. In other words, when the SVM required few iterations, LAIKA required

C	1	5	10
Australian	4	7	8
Diabetes	6	11	16
Heart	1	2	3
Adult1	17	26	34
Adult4	134	226	360

Table 4.11: SVM. Training times (in minutes). The SVM was trained with a Gaussian kernel, with width σ_{JDH}.

C	1	5	10
Australian	12	20	20
Diabetes	10	26	43
Heart	1	4	4
Adult1	45	61	51
Adult4	501	694	842

Table 4.12: LAIKA. Training times (in minutes). The SVM was trained with a Gaussian kernel, with the width parameter updated by LAIKA version 2.2.

about 50% more iterations. When the SVM required many iterations, LAIKA required about 20% fewer iterations.

The training time of LAIKA could be reduced by implementing a more efficient update strategy. The strategy implemented here was simply to clear the cached columns of the Hessian Q and recalculate from scratch as required. A better strategy would be to update the cached calculations using a power transformation (Lee and Lin, 2000).

The motivation behind the development of LAIKA was that the heuristic value σ_{JDH} underestimated the optimal value of σ when the classes were overlapping. It was suggested that this led to a high number of support vectors and a poor generalization performance. The final value of σ obtained when using LAIKA was on average about 50% higher than σ_{JDH}. It was lower in only one case, and unchanged in three cases. A search over both σ and C using the software package LOOMS (Lee and Lin, 2000) did not lead to better performance than LAIKA. The estimate of σ from LOOMS was highly variable. The estimate from LAIKA was more stable with respect to changes in the training data.

The percentage reductions in the number of SVs obtained when using LAIKA are shown in Table 4.13. On the Australian, Diabetes and Heart data sets, there was, as expected, a reduction of up to 26% in the number of SVs. On the Adult data sets there were only negligible differences between LAIKA and the SVM. In general, the number of SVs was reduced in proportion to the increase in σ (Pearson's $\rho = 0.92$).

In summary, the LAIKA heuristic estimated σ to be higher than that suggested by the original

C	1	5	10
Australian	26	22	17
Diabetes	15	18	13
Heart	0	17	14
Adult1	1	0	-4
Adult4	2	3	0

Table 4.13: LAIKA. Percentage reduction in number of SVs.

heuristic. This increase led to a more smoothed decision surface that correspondingly had a simpler description in terms of the number of support vectors. This smoother decision surface had a slightly better generalization performance. The improvement was more marked when C was high, i.e. when there was less explicit regularization. It is debatable whether these improvements justify the longer training times, although these could be reduced by use of more efficient algorithmics.

4.3 Inducing Sparsity

When learning complex decision functions, and in noisy domains, the number of support vectors can be very high. Osuna et al. (1997a) give an example of a data set with 110000 points that generates 100000 support vectors. This leads to a complex model that is difficult to interpret and can be "abysmally slow in test phase" (Burges, 1998). As for other classification algorithms, when the data are non-separable the SVM solution is a trade-off between smoothness of the decision function and training error. The resultant set of SVs contains *all* training errors, and these data points make the largest contribution to the decision function. This counterintuitive result is an indirect result of the way that the optimization problem, (3.15) or (3.22), is specified. Heuristics for constructing a decision function to which the training errors contribute less can be incorporated into the SVM to provide a sparser solution with little or no loss in generalization performance.

In Section 4.3.1, I describe a heuristic for data cleaning introduced by Burbidge et al. (2001b). In Section 4.3.2, I describe the data sets used and the experimental set-up, followed by performance results and a discussion.

4.3.1 Data Cleaning

In the support vector machine solution all training points $\mathbf{x}_i$ with $\alpha_i > 0$ appear in the solution. This includes *bounded* support vectors, for which $\alpha_i = C,\ 0 < \xi_i < 1$, which lie within the margin band (also termed *margin errors*), and training errors, for which $\alpha_i = C,\ \xi_i > 1$. Intuitively, it is not desirable for training errors to appear in the solution, especially not with maximal weight.

Training errors appear in the expansion (3.21) since $\alpha_i = C$ for points $\mathbf{x}_i$ that are misclassified. Intuitively, we would prefer a decision function that did not depend on noisy training examples. Also, if the data are noisy, or the classes overlap, the SVM solution will have many SVs. This removes one of the oft-cited advantages of SVMs — sparsity of the model. A sparse model is preferable for two reasons: it is easier to interpret and quicker to evaluate.

A simple method of excluding noisy points from the decision function is to run a standard SVM on the training data, to obtain an intitial solution. Training errors are identified as noisy points and removed. The algorithm is subsequently retrained on the reduced data. This method was called RaR (Remove and Re-train) in Burbidge et al. (2001b). On the data sets analysed, RaR provided a sparser model than the SVM with no significant drop in test accuracy. However, the total training time was around 50% more than the standard SVM.

In analogy with LAIKA (see Section 4.2), an online heuristic to approximate this is as follows. If a point is consecutively misclassified for a set number, h_e, of subproblem iterations, then it is likely to be a training error at the global optimum. Such points are removed during training. This algorithm was called STAR (Sparsity Through Automated Rejection). Burbidge et al. (2001b) showed that STAR was quicker to train than the SVM. The update frequency h_e controlled the trade-off between quality and sparsity.

In order to effectively remove an example from the training set the gradient of the Hessian

C	1	5	10	1000
australian	0.1493	0.1478	0.1478	0.1565
diabetes	0.2250	0.2329	0.2368	0.2526
heart	0.1741	0.1852	0.1815	0.2037
adult1	0.1788	0.1869	0.1850	0.2288
adult4	0.1674	**0.1444**	**0.1527**	0.2238
p	0.1403	0.0002	0.0012	0.0114

Table 4.14: STAR. 10 fold CV error rates. The SVM was trained with a Gaussian kernel, with $\sigma = \sqrt{d/2}$, and $h_e = 100$. Error rates significantly different, at 95%, from the standard SVM are emphasized (cf. Table 4.1). The last row shows the p-values for the two-tailed t-test of the null hypothesis that STAR and the SVM have the same error rate over the five data sets.

w.r.t. that example is set to be $-\infty$ (Lee and Lin, 2000). This is an efficient way to ensure that this example is never selected by the working set selection routine of Hsu and Lin (2002). This also ensures that the termination criteria are trivially satisfied w.r.t. such points. Thus no reordering of the data or of the cached computations is required.

In the following section, I compare STAR with the standard SVM on five real-world data sets[5].

4.3.2 Results and Discussion (STAR)

The data sets used in this comparison are the Australian, Diabetes, Heart and Adult data sets described previously.

The range of each data set attribute is scaled to $[-1, +1]$. Gaussian kernels, $K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/d)$, are used. A point was removed if it had been misclassified for the last $h_e = 100$ iterations.

The generalization error rate was estimated by ten-fold cross-validation. The 10CV error rates are shown in Table 4.14. These can be compared with the results of the standard SVM, as reported by Hsu and Lin (2002), and reproduced in Table 4.1. The standard SVM has a lower CV error in half of the 20 experiments. However, if $\varepsilon^l_{\alpha\xi}$ were used to select the optimal value of C for STAR[6], then the standard SVM wins on all five data sets[7] (although the difference is only significant, at 95%, for Diabetes). This again highlights the danger of combining heuristics. If the optimal value of C could somehow be selected, then STAR would win on three of the five data sets (the only significant difference, at 95%, being on Adult4).

The performance of STAR relative to the SVM does not seem to be related to the size or dimensionality of the data sets. Without further experiments, it is not possible to say *a priori* whether or not STAR would lead to a deterioration or improvement in generalization accuracy over the standard SVM.

The number of support vectors (N_{SV}) and the number of iterations (N_{iter}) are shown in Table 4.15. These can be compared with the results of the standard SVM, as reported by Hsu and Lin (2002), and reproduced in Table 4.3. STAR never increased the number of support vectors. When there were few iterations STAR had no effect. In these cases the update time h_e could be decreased. The reduction in N_{SV} varied between 9% and 84%

[5]The performance of STAR is better than the results of Burbidge et al. (2001b), since in that work there were some minor bugs in the calculation of the ξ_i and in the updating of the cache when examples were removed. The effect of these was that the optimization algorithm terminated by circling before the global optimum was attained.

[6]In the calculation of $\varepsilon^l_{\alpha\xi}$, removed points are assumed to be LOO errors.

[7]If we assume that $\varepsilon^l_{\alpha\xi}$ always picks the optimal C for the SVM, which is reasonable given the published results and our experience.

		$C = 1$		$C = 1000$	
Data	l	N_{SV}	N_{iter}	N_{SV}	N_{iter}
australian	690	192	202	150	13259
diabetes	768	447	132	110	19116
heart	270	130	47	101	1484
adult1	1605	192	1005	593	16787
adult4	4781	287	1777	1546	100000[a]

[a]Terminated Early.

Table 4.15: STAR. The number of SVs (N_{SV}) in the solutions when using STAR is never more than that of using the standard SVM and was up to 84% lower. The number of iterations (N_{iter}) was in general increased.

The number of iterations required was in general increased. This increase was roughly in proportion with the reduction in the number of support vectors.

Overall, the generalization accuracy of STAR was the same as that of the standard SVM. However, $\varepsilon^l_{\alpha\xi}$ should be used with caution for selecting the optimal value of C. STAR reduced the number of support vectors by up to 84%, which would save considerable time during classification phase. This required an increase in training time roughly in proportion with the reduction in the number of SVs, up to twice as long as the standard SVM. Thus when classification time is more important than training time, STAR offers computational advantages with little or no loss in performance.

4.4 Summary

The SVM is a powerful and increasingly popular machine for classifying data. It has been argued that there is room for improvement. Training can be halted before the global optimum of the QP is attained, with little change in the resultant hypothesis. The width parameter of a Gaussian kernel can be automatically tuned to improve performance and avoid model order selection. Training errors can be removed during training to simplify the solution, with little change in performance.

The main aspects of these three heuristics, the contributions presented in this chapter, and some suggestions for future work are summarized below.

HERMES is a stopping criterion for the SVM based on an approximate estimate of the generalization performance. It is designed to reduce training time without deteriorating performance. Early stopping to obtain a rough idea of the SVM's performance may be useful during parameter selection and exploratory data analysis.

- Terminating training with HERMES can lead to a considerable reduction in training time.
- The accuracy of the solution was almost the same as that given by the global optimum of the QP.
- The number of training errors, and hence SVs, was increased.
- HERMES can be used to halt the SVM when other heuristics are used which are not guaranteed to converge.
- There is a slight risk that the SVM will terminate very early, which could result in a poor solution.

Some possibilities for further work include the following.

- To avoid very early stopping, only check the HERMES criterion once the KKT conditions have been met to a certain precision, say $\epsilon = 1$.
- Instead of insisting that $\varepsilon^l_{\alpha\xi}$ remain constant, allow some finite tolerance.
- Investigate the relationship between the number of examples, the dimensionality, the noise level, the working set size and the value of persistence. Ideally a default value would be supplied by some rule-of-thumb.
- It has been argued above that early stopping is useful during exploratory data analysis. To verify this claim or otherwise it would necessary to compare parameter selection with and without HERMES. In particular Lee and Lin (2000) demonstrate that solutions with $\epsilon = 0.1$ in the KKT conditions are good enough for parameter selection on many public domain data sets.

LAIKA is a heuristic to adjust the width σ of a Gaussian kernel during training. The width is adjusted to be on the scale of separation of the support vectors, in analogy with heuristics for Gaussian RBF networks. It is designed to avoid the need for model order selection and to improve accuracy over using a fixed heuristic value of σ.

- Adjusting σ to be data-dependent led to an increase in its value over the initial estimate, as expected.
- This resulted in solutions with fewer SVs, and thus shorter classification time.
- LAIKA is not guaranteed to converge. HERMES can be used as the stopping criterion.
- The accuracy of the SVM was not significantly improved by LAIKA. Adjusting σ compensated for a poor choice of C.
- $\varepsilon^l_{\alpha\xi}$ was not reliable for selecting C, but note that it does not hold for such solutions.
- Any gains in accuracy and sparsity may not be worth the increase in training time.
- The estimates $\hat{\sigma}$ from LAIKA were much more stable over different data samples than those of LOOMS.
- The performance was barely affected by the update period h_a (provided $h_a \ll N_{\text{iter}}$).

Some possibilities for further work include the following.

- In cases where $h_a \geq N_{\text{iter}}$, update σ after training and repeat training with $h_a \leftarrow h_a/2$, say.
- Investigate the effect of h_a on training time, with respect to the number of examples and dimensionality. As for the other heuristics, it would be desirable to have a rule-of-thumb for this parameter.
- A larger σ increases regularization, which implies that a higher C is required than for the initial value σ_{JDH}. This suggests investigating the optimal C value for the SVM with σ_{JDH} and with LAIKA and whether or not this is quantifiable.
- Similarly to Chapelle et al. (2002), a different σ^p could be introduced for each feature. Adaptive scaling of these could lead to improved performance and feature selection. (This has shown some promise on early trials, but a heuristic that accounts for interactions between features is probably required.)

- A similar heuristic could be developed for other kernels, for example the polynomial $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^t\mathbf{z}/\sigma + \tau)^d$. (For a linear kernel, $\tau = 0$, $d = 1$, this is equivalent to automatically tuning C. On the pK_i data with MACCS keys, this usually led to improved performance and a sparser model. The results are not reported in this thesis.)
- The recalculation of the cache could be made more computationally efficient (Lee and Lin, 2000).
- As for the other heuristics, it may be preferable to only update σ when the SVM is near convergence, e.g. when the KKT conditions are satisfied to $\epsilon = 1$.
- In analogy with RBF networks, it is possible to allow a different σ_i for each SV. This should result in a sparser solution. The kernel function is then not guaranteed to be positive definite, other optimization techniques could be used (Mangasarian, 2000).

STAR is a heuristic to remove potential training errors during training. It is designed to result in a solution with the same predictive performance as the SVM with fewer SVs. Such a solution is quicker to evaluate and may be easier to interpret.

- The solutions provided by STAR had very similar accuracy to the SVM.
- The number of SVs was reduced, considerably in some cases.
- The number of iterations was increased.
- $\varepsilon^l_{\alpha\xi}$ was not reliable for selecting C. Note that it is not valid for such solutions.

Some possibilities for further work include the following.

- It would be interesting to investigate whether or not the removed points are training errors, or SVs, in the original SVM solution and whether or not they are misclassified by the STAR solution.
- The optimal update period h_e is likely to depend on the training set size and dimensionality and the amount of noise. Removing points too soon may be over-zealous.
- When $h_e \geq N_{\text{iter}}$, remove the training errors (cf. RaR, Burbidge et al., 2001) and retrain with $h_e \leftarrow h_e/2$, say.
- For larger data sets, STAR may remove points too soon. As for the other heuristics, it may be preferable to invoke STAR only once the KKT conditions have been satisfied to some loose precision.
- The behaviour of STAR has only been investigated for the Gaussian kernel. Its effect on the SVM with other kernels is of interest. STAR is used with a linear kernel in Chapter 5.
- Remove all margin errors ($\xi_i > 0$) as well as training errors ($\xi_i > 1$). This should further reduce the number of SVs. (Early trials have been inconclusive.)
- The principle behind STAR could be applied to other classification, or even regression, algorithms. That is, use a learning machine as a wrapper to remove spurious examples from the training data before training. A fast algorithm could be used as a filter for a more sophisticated, slower algorithm.
- Analysis of the proportion of removed examples, in comparison to the proportion of SVs may suggest further avenues.

- Since $\varepsilon^l_{\alpha\xi}$ is qualitatively the same as the test error before convergence, it could be used to remove likely leave-one-out errors during training.

All of the heuristics achieved their aims to some extent. They should prove useful in the classification of drug discovery data. This is investigated for the pK_i data in the following chapter. The heuristics also have wider applicability to the SVM and to other learning machines.

Chapter 5

Application to pK_i Data

If your experiment requires statistics you should have designed a better experiment.

ERNEST RUTHERFORD

In Chapter 3 (Section 3.3.2) the pK_i data were described. There were activity data for 1415 compounds (ligands) which had been screened against 11 targets (receptors) in competitive binding assays. The activities were reported as pK_i values, which are directly comparable across different screens. The aim was to learn a classification rule to discriminate between 'active' and 'inactive' compounds, where 'active' means having a pK_i value greater than 6. This is a pragmatic threshold set by the screening group.

It was found that, on the basis of 14 physicochemical and one-dimensional structural properties, the standard SVM with a Gaussian kernel did not provide a significantly higher accuracy than a decision tree. In Section 5.1, two methods to improve the SVM's performance are considered. These are selecting the training data by clustering in the space of fingerprints, and adjusting the misclassification costs according to the class priors.

It was suggested that the SVM was overfitting the training data. This is possibly because the width of the Gaussian kernel was underestimated. This was also suggested by the high number of support vectors. These observations, and similar observations from analysis of public domain data sets, motivated the development of the heuristics of the previous chapter. In Section 5.2, these heuristics are used in the analysis of the pK_i data. Note that although the heuristics have already been presented in their 'finished' form, they were actually developed concurrently with the data analysis. Thus, the early versions of STAR and LAIKA were first applied to the analysis of the pK_i data. They were found wanting, and subsequently further developed on public domain data sets as described in the previous chapter. These improved versions were then reapplied to the analysis of the pK_i data. This process will have introduced some bias into the analysis ('data snooping'). The earlier results are not reported as the experimental set-up varied considerably.

A further suggestion in Chapter 3 (Section 3.3.2.2) was to use an alternative representation of the compounds. Three alternative representations are used to learn a classifier. In Section 5.3, the 'finished' heuristics are also applied to these classification problems. Although the compounds are the same as before, from a classification perspective the data sets are very different. Thus there should be minimal 'data snooping' bias in these later analyses.

In Section 5.4, the performance of the SVM and the heuristics for the various experimental set-ups is summarized. These suggest further analysis and possible improvements to the heuristics.

5.1 Some Data Mining Considerations

Before considering the application of the improved heuristics to the analysis of these data, it is useful to consider some generic methods to improve the performance. In Section 5.1.1, the problem of selecting training data is considered. A clustering algorithm, popular in chemoinformatics, is applied to this problem. This also highlights some benefits of exploratory data analysis. In Section 5.1.2, the problem of maximizing relative advantage (RA) is considered. Minimizing a risk functional is chosen as a proxy to this. However, it is shown that minimizing expected error leads to a higher RA, for these data. Which method should be used for selecting training data, and which risk functional should be used depend on the requirements of the analyst. Some pragmatic issues are discussed in these sections. In Section 5.1.3, the relationships between the performance measures and between the performance measures and their estimates are summarized. In Section 5.1.4, these considerations are summarized.

As in Chapter 3 (Section 3.3.2), the SVM was trained with a Gaussian kernel with width σ_{JDH}. For a fair comparison with the heuristics later in this chapter, the training was terminated by HERMES with a persistence of 1000. The SVM was trained for $C \in \{1, 10\}$.

5.1.1 Training Data Selection

Obviously, increasing the amount of training data will always improve the performance of a classifier (given that the i.i.d. assumption holds). Obtaining this data is expensive though as more compounds must be screened. More training data also means longer training time, so smaller, more informative training sets are preferable. One possibility is to attempt to identify in advance which as yet unscreened compounds would be most informative as training data were they to be screened. Fung and Mangasarian (1999) use k-mediod clustering to select a subset of the available (unlabelled) data to be labelled and used to train an SVM. This is analogous to using clustering to determine the centres of an RBF network (Bishop, 1995).

A similar approach is taken here, incorporating domain knowledge. The compounds were clustered using the clustering of Ward (1963) (see also, Chapter 2, Section 2.3.3) in the space of Daylight fingerprints (Chapter 2, Section 2.2.3). The Tanimoto measure was used as the measure of similarity. The clustering is controlled by a similarity threshold. At a threshold of 0.7 this led to 395 clusters[1]. The 'centroid' of each cluster is included in the training set. This is not a true centroid, since the Tanimoto measure does not define a metric space. Also, the bit string most Tanimoto-similar to the cluster members may not correspond to an actual compound. That compound is chosen which is most similar to the other cluster members (McGregor and Pallai, 1997).

If the training data are chosen thus, it is hoped that the resulting classifier will be sparser and more accurate, since the training data should be more representative and more informative. Since there is only a finite amount of data, the SVM is trained on the centroids and tested on the remainder (the analogues). The training and test data no longer satisfy the i.i.d. condition on which the SVM methodology and error bounds are based.

The clustering revealed an aspect of the data that could have biased the results. There were 150 compounds in the analogues set that were identical to their respective cluster centroids with respect to the fourteen properties used. That is, although all the compounds were distinct physically, there were pairs of compounds indistinguishable on the basis of the features used. Visual inspection of the structures of some of these revealed that they were chiral pairs (enantiomers of each other).

[1] Hence, this was chosen as the size of the random training set, for the purposes of comparison.

That is, the only difference between such a pair was the orientation of the substructures on a particular carbon atom.

When training on the random data set, there are three possibilities for such a pair:

1. Both compounds are in the training set. If they have the same label (i.e. both active or both inactive) this amounts to a duplication of that datum. If they are labelled differently, the training data are inconsistent, hence there will necessarily be non-zero training error. Which enantiomer is mislabelled by the trained SVM will depend on the other training data.

2. Both compounds are in the test set. The above argument then applies to the test set, so there will be non-zero test error if the compounds are labelled differently.

3. One compound is in the training set and one is in the test set. If they are labelled the same, this should lead to an increased probability of a correct prediction, and conversely, a contradictory labelling will probably lead to a decrease in predictive performance.

When training on the centroids, only the third case above can hold. Since the chiral pairs are more likely *a priori* to be labelled the same (by the similar property principle) the repetitions should lead to a net increase in predictive performance in this case. This is what was observed. However, how much of this is due to the presence of chiral pairs and how much is due to an information gain from the clustering *per se* is impossible to say without further analysis.

The remaining question is: what does the analyst do in light of this new information? For a given chiral pair, if both enantiomers are in the training set, an immediate response may be to remove the duplicates. There are two possibilities for a given pair:

1. If the enantiomers have the same label, then removing one of them will skew the class priors and the class densities.

2. If the enantiomers have contradictory labels, which one should be removed? Pragmatically, adjusting the activity threshold may reduce inconsistencies. Alternatively, manual inspection may lead a medicinal chemist to treat them both as active or both as inactive, or remove them both. These approaches require valuable person time.

A more attractive alternative is to leave both points in the training set, so as to not disrupt the densities, with a prior ignorance about the label. This could be achieved by a transductive SVM (Vapnik, 1998; Joachims, 1999b). An approximation is to leave both in with their opposing labels, and let the SVM determine which one falls on the wrong side of the hyperplane, according to the distribution of the other points.

If one or both points are in the test set, then in a practical situation, we do not know whether the labels are inconsistent. Hence, the best we can do is leave them both in and hope that the descriptors are sufficient to discriminate between them. In this case, we have ground truth data, so we know that the descriptors are insufficient.

Thus, in the remainder, all of the available compounds are used when learning a classifier, and the test data are assumed *a priori* to be consistent.

5.1.1.1 Performance Comparison

Since there are only 1415 compounds for which activity is known, comparing the performance of different predictive methods is compromised in this case. The two methods under consideration involve different selection criteria for the training data. In the case of random selection, the training

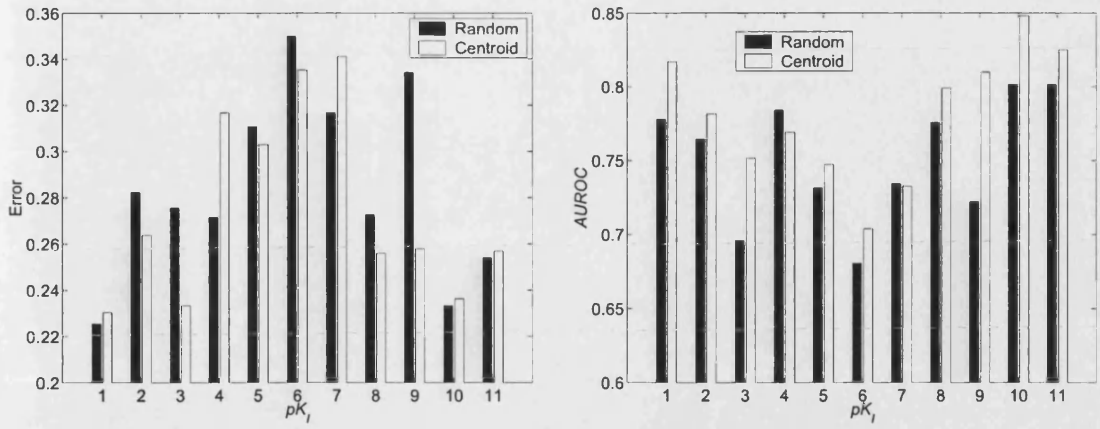

Figure 5.1: pK_i. The error (*left*) of the SVM ($C = 10$) when training on the centroids is higher on five assays. However, the $AUROC$ (*right*) is approximately the same or higher on ten assays, which suggests the operating point has not been set optimally.

and test data are i.i.d. from the same underlying distribution. Thus the error rate on the test set is an unbiased estimate of the true error rate. In the case of centroid selection, the training and test sets are not i.i.d. and have different distributions. The test error is thus a biased estimate of the true error rate. Since the average distance from a test to training point is greater in the case of centroid selection than in the case of random selection, we expect the test error to be biased upward. Moreover, since the test sets for the two methods are drawn from different distributions, the test error rates are not directly comparable. In the following, I report the test error and the AUROC and discuss the effects of this bias. Since the error rates are not comparable, standard errors and p-values are not reported. The results presented are merely suggestive. Note that the non-parametric McNemar test for comparing two classifiers is not applicable as the test sets differ.

If we were only interested in estimating the performance on this set of 1415 compounds, then the performance metric would be the overall error on the training and test sets. In this scenario, this measure would be directly comparable for the two methods.

There are two possibilities for obtaining a fair comparison under more general conditions. One would be to use a third set of compounds, drawn from the same distribution, as an independent test set. There are a few thousand compounds available whichh have been screened on assay 3 for which this is possible. Another would be to use cross-validation. For example, split the data into five folds of 283 compounds. For each fold, choose the training data by clustering the remaining 1132 compounds and selecting the centroids. Choose the random training set from the 1132 compounds to be of the same size. Train the SVM on each training set and test on the left-out fold. Repeat this for all five folds and average the error rates. This provides an unbiased estimate and makes use of all the data in estimating the true error rate. However, it is time-consuming and was impractical for these data given the available hardware.

5.1.1.2 Results and Discussion

The misclassification rates on the test set are illustrated in the left of Figure 5.1, for $C = 10$. (The results for $C = 1$ are similar.) In the right of Figure 5.1 is illustrated the corresponding area under the ROC curve ($AUROC$, Chapter 3, Section 3.2.1)). When using the centroid training data, the test error is increased on five assays. However, on three of those assays the $AUROC$ is increased. For only two assays (4 and 7) does the $AUROC$ decrease, although not by much.

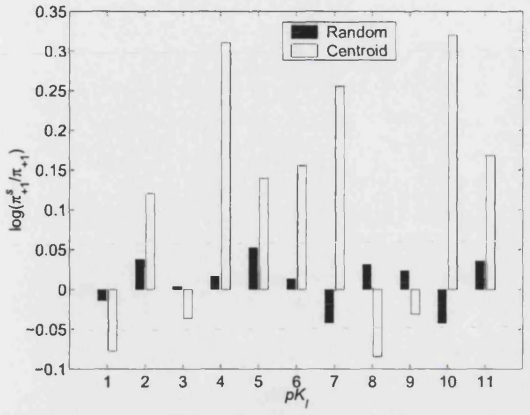

Figure 5.2: pK_i. The relative differences between the (test) population priors, π_{+1}, and the (training) sample priors, π^s_{+1}, are larger when the sample data are selected by clustering. This leads to the operating point being set sub-optimally (cf. Figure 5.1).

This suggests that the operating point b in the SVM solution has not been set optimally. The reason for this becomes evident when one considers the class priors. When the training data are randomly selected, the proportion of active compounds in the future will on average be the same as that in the training data. When the training data are selected by clustering, it is possible that the proportion of positives in the training data will differ from that in the population. This can be seen in Figure 5.2. The proportion of compounds in the population (i.e. the test set), π_{+1}, deviates from that of the training set, π^s_{+1}, the most on assays 4,7,10 and 11. From Figure 5.1 we can see that on these assays the test error increased. However, on assays 10 and 11 the *AUROC* increased and on assay 7 it remained approximately the same. On assay 4 *AUROC* only decreased slightly whereas the test error increased considerably. Thus it is probable that the ROC curve could be used *post hoc* to adjust the operating point to reduce the test error. Alternatively, if the true population proportions were known in advance then the correction to the regularization constants for the two classes, C^+ and C^-, can be made (see Chapter 3, Section 3.1.4). On the two assays for which the proportion of positives in the analogues (test) set was within 5% of that of the centroids (training) set, viz. assays 3 and 9, there was a substantial reduction in error rate and increase in *AUROC*. This suggests that if the true priors were known and the corresponding correction made then training on the centroids would give superior predictive performance to training on the randomly selected data.

The relative advantage (*RA*, Chapter 3, Section 3.2.2) was on average 2% higher when using the randomly selected training data.

The number of support vectors was about 4% higher on average when training on the centroids, although the increase was as much as 20% in come cases. The number of iterations was about 5% higher on average when training on the centroids, although it varied considerably across assays.

Which method should be used to select the training data therefore depends on the properties desired of the learned decision rule. If one is designing a tool for screening set selection then the randomly selected training data would be better for two reasons. Firstly, the solutions are sparser, which leads to a shorter classification time. This is important when there are potentially millions of compounds in a virtual library. Secondly, *RA* is slightly increased, i.e. there is an increased gain over randomly selecting compounds for *in vitro* screening.[2] A further reason for preferring to use

[2]Random selection of training data for learning a rule should not be confused with randomly selecting compounds for *in vitro* screening.

	Random		Centroid	
	$C = 1$	$C = 10^a$	$C = 1$	$C = 10^b$
AUROC	—	—	—	1%
RA	−4%	−2%	−1%	−2%
N_{SV}	+4%	+1%	+5%	+5%
N_{iter}	−8%	+32%	−8%	+35%

[a]Assays 6 and 7 excluded.
[b]Assays 1,3, and 7.

Table 5.1: pK_i. Training the SVM with weighted misclassification costs leads to a slight decrease in relative advantage (*RA*) and a slight increase in the number of support vectors (N_{SV}). The number of iterations was decreased when $C = 1$ and increased when $C = 10$. *AUROC* is almost unchanged.

the randomly selected training data *per se* is that the assumption that the population is i.i.d. as the sample then holds. This is a key assumption underlying statistical learning theory and the corresponding algorithms and error bounds.

The effect of using clustering to select the training data is reconsidered in the following section with respect to a different loss function. The effect of using the fingerprints for clustering and MACCS keys for prediction is analysed in Section 5.3.1.1. Unless otherwise stated, the remainder of the results in this chapter are for randomly selected training data.

5.1.2 Misclassification Costs

In an effort to improve the relative advantage of the classifier, the SVM was retrained using the loss ratio $\lambda(-1,+1)/\lambda(+1,-1) = \pi^s_{-1}/\pi^s_{+1}$. This is equivalent to minimizing the false positive rate plus the false negative rate, as opposed to minimizing the error rate. This was shown empirically in Chapter 3 (Section 3.1.5.3) to maximize the relative advantage. That is, the relative increase (over random selection) in the number of actives that would be found if the compounds predicted positive *in silico* were physically screened *in vitro*.

Since the SVM is now being trained to minimize the expected risk rate as opposed to expected error, these are not sensible measures with which to compare the two approaches. The *RA* and the *AUROC* are useful performance measures for the comparison.

In a strictly probabilistic setting the *AUROC* should not depend on the loss ratio, since the ROC curve would be generated by varying the loss ratio (Hand and Till, 2001). For an hyperplane classifier, such as an SVM, an ROC curve can be generated by varying the operating point b in $f(\mathbf{x}) = \text{sgn}(\langle\phi(\mathbf{x}), \mathbf{w}\rangle_{\mathcal{H}} + b)$. This does not give the same ROC curve as varying the loss ratio and retraining each time. Even if the *AUROC* is similar for two different loss ratios, the shape of the curve is likely to be different. In practice, the *AUROC* was almost unchanged. The change in *AUROC*, *RA*, number of support vectors (N_{SV}) and number of iterations (N_{iter}) are presented in Table 5.1. The experiments with weighted costs chronologically preceeded those with equal costs. The maximum number of iterations for the former set was set at 5000, which was in some cases too few for the centroid data when $C = 10$. Results are averaged over those assays for which the SVM converged within 5000 iterations for both loss functionals.)

There was a slight deterioration in *RA* and N_{SV}. For $C = 1$ the number of iterations was decreased. However, the case $C = 10$ takes roughly four times as many iterations. Training with weighted costs increased this by about a third. Overall, there is a slight deterioriation in performance, and time- and space-complexity when training the SVM with weighted misclassification costs on these data.

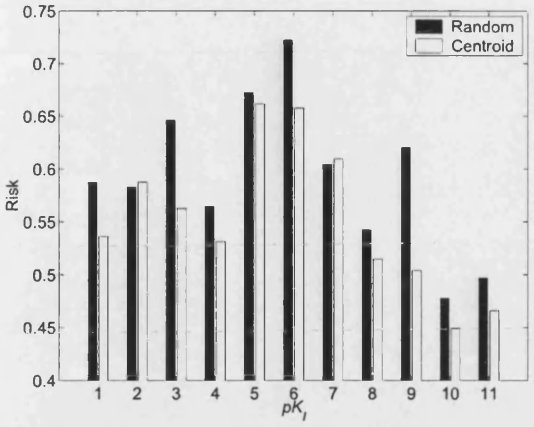

Figure 5.3: pK_i. When using weighted losses the risk is reduced by training on the centroids.

5.1.2.1 Random vs Centroids

When minimizing the error rate, training on centroids was found to result in a higher $AUROC$ at the expense of a slight increase in the number of SVs and the number of iterations. However, the error rate was not always improved, possibly due to the skewing of the class priors and hence the operating point (Section 5.1.1.2). The comments of Section 5.1.1.1 concerning performance comparison also apply to the following discussion.

When minimizing the weighted loss functional, the results for $C = 1$ are as follows. (When $C = 10$, the SVM converged within 5000 iterations for both methods on only two assays.) $AUROC$ was slightly increased by training on the centroids. The risk of using the SVM was correspondingly decreased on nine assays and was almost the same on the other two (Figure 5.3). Minimizing this risk functional was chosen to maximize RA. However, the RA was almost the same for either method of selecting training data.

The number of SVs when training on the centroids was again on average 4% higher when $C = 1$. It was unchanged when $C = 10$ for those cases where the SVM converged within 5000 iterations. The number of iterations when training on the centroids was on average 12% lower when $C = 1$. However, when $C = 10$, the SVM trained on the centroids did not converge within 5000 iterations on eight assays, and was on average 27% slower on the others.

In terms of predictive performance, selecting the training data by clustering is preferable. For screening set selection, selecting data randomly is slightly preferable, as the solution is quicker to evaluate (less SVs) and has the same RA.

5.1.3 Meta-Analysis

In the preceding analysis, the SVM has been trained to minimize two loss functionals. The aim has been to maximize RA and $AUROC$. In Section 5.1.3.1, the relation between these four measures is investigated. In Section 5.1.3.2, an estimator of the loss is compared to the losses incurred on the test data.

5.1.3.1 Relationship Between Performance Measures

The predictive performance of the SVM has been evaluated with a variety of measures. The $AUROC$ summarizes in a single number the risk of using the SVM over a range of misclassification costs. The RA is the enrichment of actives in a set selected for screening by the SVM, compared to

Measure	Loss	Random		Centroid	
		$C=1$	$C=10$	$C=1$	$C=10$
$AUROC$	error	-0.80	-0.77	-0.83	-0.77
$AUROC$	risk	-0.99	-0.85^a	-0.99	-0.97^b
RA^{-1}	error	0.74	0.74	0.70	0.42
RA^{-1}	risk	0.83	0.83^a	0.66	0.94^b

[a]Excluding assays 6 and 7.
[b]Assays 1,3 and 7.

Table 5.2: pK_i. Correlations between performance measures, when minimizing the specified loss functional.

randomly selecting compounds for screening. The SVM is trained to minimize a (regularized) loss functional. It is not possible to specify a loss functional that directly maximizes either $AUROC$ or RA. It would be hoped that a low error rate (or risk) corresponds to a high $AUROC$ or RA. A loss functional was introduced in Chapter 3 (Section 3.1.5.3) that should vary approximately inversely with RA.

To give a rough idea of how these measures are interrelated, their linear correlations are given in Table 5.2. The error and risk (when each is respectively minimized) are highly correlated with $AUROC$. The risk is more strongly correlated. (Recall that the $AUROC$ was almost the same for the two loss functionals.) The RA was inversely correlated with error and risk. It was more strongly correlated with risk. However, minimizing error rate led to higher RA than minimizing risk. Since there are only 11 data supporting each entry in the table, the correlations should not be taken too seriously. The patterns seem fairly consistent for the different values of C and different training sets.

5.1.3.2 Behaviour of Error Estimators

The main purpose of the work presented in this chapter is to evaluate the effect of various methods for improving the solution of the SVM. However, given any one set-up, e.g. equal misclassification costs, randomly selected training data, SVM with Gaussian kernel, it is necessary to set the regularization parameter C. Although this problem is not discussed at length in this work, it is useful to briefly consider some simple methods. Sollich (1999, 2000) presents a probabilistic argument that suggests C should not be too small ($C \geq 1$, roughly). Cristianini and Shawe-Taylor (2000) also present a result that suggests $C = R^{-2}$, where R is the radius of the data. For a Gaussian kernel this implies $C \geq 1$. The experiments reported here have used $C \in \{1, 10\}$ only, which seems a reasonable, if scanty, range.

One way to set a free parameter of a learning machine is to train it for a range of values of that parameter and use that value minimizing some estimate of the performance. For the SVM, the expected error rate, or risk for the case of unequal misclassification costs, can be estimated in a number of ways. Since the SVM is a regularized learning machine, one would hope that the training risk[3] could be used as an estimate of the expected risk. However, this is only the case for the optimal C, so this method is unlikely to succeed for selecting C. For the four set-ups reported above, the training risk was always smaller for $C = 10$. This is as expected, since $C = 10$ imposes a higher penalty on training risk.

As noted in Chapter 3 (Section 3.1.5.2), the leave-one-out (LOO) risk is an almost unbiased estimator of expected risk. This can be estimated by $\varepsilon^l_{\alpha\xi}$. This estimator is strictly valid only at the global optimum of the quadratic program. However, it was shown in Chapter 4 (Section 4.1.2),

[3]In the following 'risk' is any loss functional, including error rate.

on some public domain data sets, that before convergence $\varepsilon^l_{\alpha\xi}$ is qualitatively the same as the test error. Thus it may be a useful estimator of risk even when the SVM is terminated by HERMES or after a set number of iterations. For a given value of C, $\varepsilon^l_{\alpha\xi}$ correlated well ($\rho \in [0.90, 0.94]$) with test error or test risk over the 11 assays when using randomly selected training data. It is also only valid if the test data are i.i.d. as the training data, which is not the case when the training data are selected by clustering. In this case, $\varepsilon^l_{\alpha\xi}$ correlated less well ($\rho \in [0.58, 0.84]$) with test error or test risk. However, for a given assay and set-up, $\varepsilon^l_{\alpha\xi}$ was smaller for the better value of C in only 19 out of 44 cases (five out of 11 for the cases where the assumptions behind $\varepsilon^l_{\alpha\xi}$ hold). Thus it seems that while, for a fixed C, $\varepsilon^l_{\alpha\xi}$ is qualitatively the same as the test error, it is no better than random for choosing the better value of C. This is disappointing, given the results reported in the literature (Joachims, 2000; Duan et al., 2001).

5.1.4 Summary of Data Mining Considerations

The aim of this analysis has been to learn a decision rule to predict whether or not a compound is active. The quality of such a decision rule can be evaluated in terms of its predictive accuracy and its time- and space-complexity.

The predictive accuracy can be quantified by the relative advantage (RA) of using a learned decision rule to select compounds for synthesis and screening over random selection of compounds. An alternative is to consider the misclassification rates over a range of costs, as summarized by the area under the ROC curve ($AUROC$). This latter measure is useful since the misclassification costs are subjective and difficult to quantify. It is not possible to directly maximize these quantities in the standard SVM framework. Minimizing a (possibly) weighted expected misclassification rate is a proxy for maximizing RA or $AUROC$. Minimizing a specific expected risk (false negative rate plus false positive rate) was highly correlated with maximizing RA and $AUROC$. However, minimizing the expected misclassification rate led to the same $AUROC$ and a higher RA.

It should be noted that varying the operating point b in the SVM solution is not the same as varying the loss functional and retraining. For these data, the orientation of the hyperplane varied with the loss functional. Equal $AUROC$ does not imply the classifiers are the same.

A second performance criterion is classification speed. If the decision rule is to be used to select compounds for *in vitro* screening, then it should be quick to evaluate. There are generally many more compounds whose activity we would like to predict than are screened. Thus, whilst training time should not be too long, classification time is more important. For an SVM, the classification speed is $O(N_{\mathrm{SV}})$. Solutions with few SVs are therefore preferable.

Clustering the compounds in the space of fingerprints revealed that the properties used to describe the compounds did not have sufficient resolution to identify them uniquely. This led to repetitions and inconsistencies from the viewpoint of hyperplane classification.

The $AUROC$ of the SVM was slightly increased when the clustering was used to select the training data. However, random selection is preferred in this case, since it led to slighty higher RA and slightly sparser solutions. This is a pragmatic decision.[4] Note that when minimizing risk, training on the centroids always led to a lower risk. However, this did not translate into an increase in RA. The performance of any learning machine trained on data selected by clustering will depend on the clustering algorithm, its parameters and the representation used.

Unless stated otherwise, in the remainder of this chapter, the results are reported for the case of equal misclassification costs and randomly selected training data. With this set-up, training

[4]Note that most of the results in the rest of this chapter are qualitatively the same for both methods of selecting training data.

the SVM with the physicochemical and one-dimensional structural properties as descriptors led to solutions with many SVs, that did not perform much better than a decision tree (Chapter 3, Section 3.3.2.2. In the remainder of this chapter, further attempts to improve the predictive performance and reduce the number of SVs are considered.

5.2 Evaluation of Heuristics

The heuristics STAR and LAIKA were introduced in the previous chapter. Their development was motivated in part by the results and analysis of the preceding sections. When classifying the pK_i data, the SVM gave solutions with many SVs with poor predictive performance. STAR (Chapter 4, Section 4.3) was designed to result in sparser solutions, without increasing training time or reducing predictive performance. LAIKA (Chapter 4, Section 4.2) was designed to automatically tune the 'width' σ of the Gaussian kernel, which should lead to improved accuracy and a sparser solution. In Section 5.2.1, STAR is applied to the classification of the pK_i data. In Section 5.2.2, LAIKA is applied to the classification. The compound representation used is the same set of fourteen physicochemical and one-dimensional structural properties as before.

When using the heuristics, the SVM is not guaranteed to converge to the global optimum of the quadratic program. Hence, the HERMES criterion is used is the stopping criterion with a persistence of 1000. That is, unless the global optimum of the QP was attained first, training was halted if there had been no change in $\varepsilon^l_{\alpha\xi}$ for 1000 iterations. For pragmatic reasons, the maximum number of iterations was set at 10000. The combination of STAR and LAIKA was not fully evaluated owing to excessively long training times.

5.2.1 STAR

The predictive accuracy of STAR was almost the same as the SVM across all assays, although it was very slightly lower on average when $C = 1$. The *RA* and *AUROC* of STAR were almost identical to that of the SVM across the assays. Thus, the predictive performance of STAR has not deteriorated from that of the SVM, under any of the three performance measures. The purpose of STAR though is to reduce the number of support vectors. The percentage change in the number of support vectors (N_{SV}) when using STAR is illustrated in Figure 5.4. N_{SV} is reduced on all 11 assays by about one third when $C = 1$ and by about one eighth when $C = 10$. The training time was also reduced in most cases.

In summary, STAR has predictive performance no worse than the SVM, and results in a sparser solution than the SVM, usually in a shorter runtime.

5.2.2 LAIKA

As noted in Chapter 4 (Section 4.2.3), on the pK_i data, the original version (v1.0) of LAIKA did not improve the SVM solution and required longer training times. It was proposed that this was due to the high number of support vectors (SVs) resulting in an underestimate of the kernel parameter σ (the 'width' of the RBF). On these data, LAIKA v1.0 actually led to an estimate $\hat{\sigma} < \sigma_{\mathrm{JDH}}$ and an *increase* in the number of SVs. In Chapter 4 (Sections 4.2.3 and 4.2.4) the heuristic was developed to overcome these problems in two ways. Firstly, by basing the calculation only on correctly classified SVs, the noise or overlap of the classes should not affect the estimate. Secondly, by only updating on points that have been correct SVs for a set number of iterations the

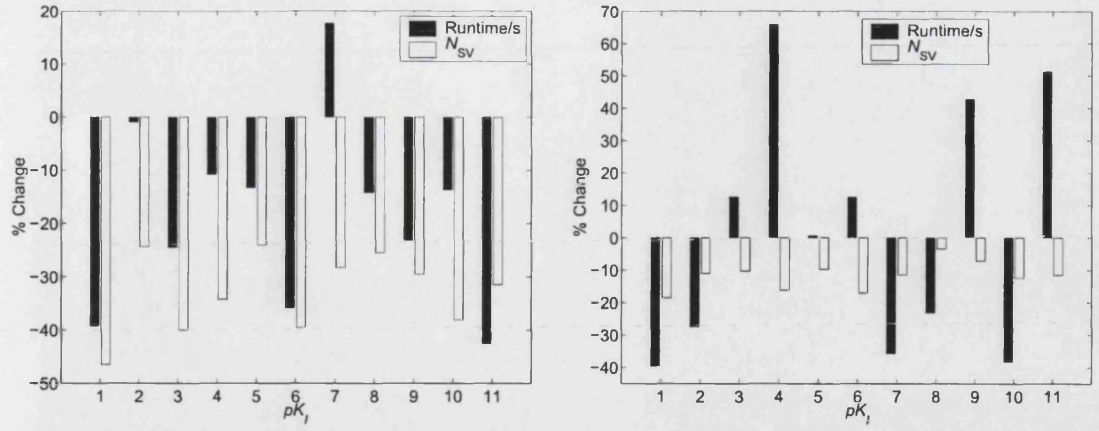

Figure 5.4: pK_i. STAR outputs a solution with fewer support vectors (N_{SV}) than the SVM both for $C = 1$ (*left*) and $C = 10$ (*right*). The training time required by STAR is in general lower than for the SVM.

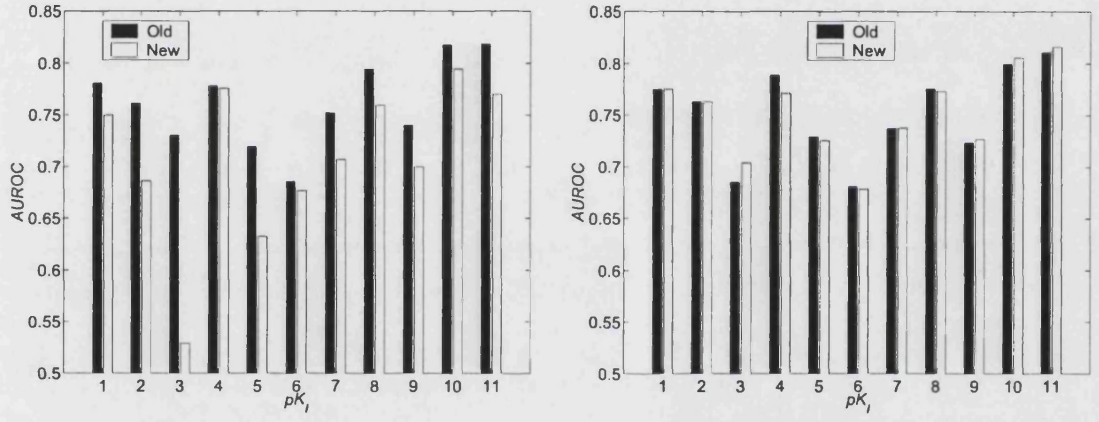

Figure 5.5: pK_i. The *AUROC* obtained by the new version of LAIKA is always lower when $C = 1$ (*left*) and approximately the same when $C = 10$ (*right*).

estimate is not as affected by transient SVs. Thus the solution should be more stable and quicker to find.

The *AUROC* obtained when using the previous version (i.e. v1.0), compared to using the new version (i.e. v2.2), is illustrated in Figure 5.5.

When $C = 1$ the *AUROC* obtained by the new version of LAIKA is always lower than that obtained previously. Note that training terminated at the global optimum on all assays except assay 2. Thus the deterioration in *AUROC* is due to LAIKA v2.2 alone. When $C = 10$ the *AUROC* is approximately the same as that obtained previously. Note that only assay 3 terminated at the global optimum. LAIKA was halted after 10000 iterations on assays 6 and 7, and was terminated by the HERMES criterion on the other assays.[5]

LAIKA v1.0 gave solutions almost identical to the SVM. Thus the preceding argument carries over to comparing LAIKA v2.2 to the SVM. That is, when $C = 1$, LAIKA v2.2 returns a solution with lower *AUROC* than the SVM (by about 7% on average). The test error is correspondingly higher and the *RA* slightly lower. On the three assays for which LAIKA required more iterations

[5]It is tempting to speculate that the solution did not deteriorate when $C = 10$ because LAIKA was halted early. That is, overfitting was prevented by early stopping (McLoone and Irwin, 2001). This is not the case, see below.

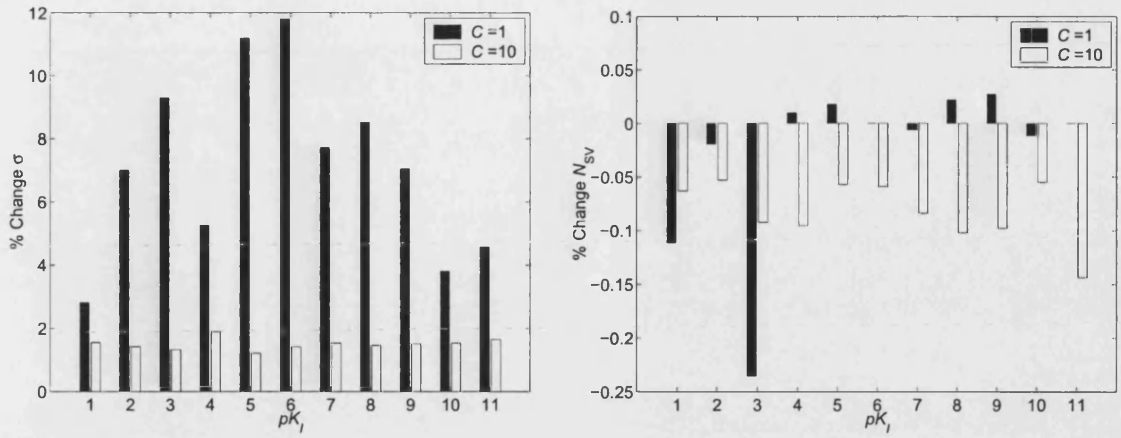

Figure 5.6: pK_i. The final value of σ (*left*) for LAIKA v2.2 when $C = 1$ is much larger than the estimate σ_{JDH}. This led to a high training error (underfitting). When $C = 10$, the final value of σ was about 50% higher, which resulted in similar predictive performance with fewer SVs (*right*).

(by a factor of three) the number of SVs was reduced the most. On the other assays, LAIKA required roughly half the training time of the SVM and the number of SVs was approximately the same. The number of iterations (and number of SVs) was not correlated with the test error. Note that the time per iteration of LAIKA was twice that of the SVM. Thus, even when the number of iterations was reduced, the training time was usually higher. As noted in Chapter 4 (Section 4.2.6), this could be improved by better algorithmics.

When $C = 10$ the performance of LAIKA v2.2 is nearly the same as the SVM, in terms of *AUROC*, *RA* and test error. The number of iterations was doubled and the training time was trebled. The only gain was a slight (5%–14%) reduction in the number of SVs.

When $C = 10$, on assays 1, 2, 4, 5, 9, 10 and 11, both the SVM and LAIKA were terminated by HERMES. Thus the speculation that early stopping prevented deterioration of the solution is unsupported. The question remains: why has the perfomance deteriorated when $C = 1$ and not when $C = 10$? The previous speculation for v1.0 was that LAIKA was overfitting. An inspection of the training and test errors shows that LAIKA was actually *underfitting* the training data. This is also apparent from the estimates of the width, $\hat{\sigma}$, obtained (Figure 5.6, *left*). When $C = 1$, the estimate of σ is too high, thus the decision surface is too smooth to capture the inherent complexity of the data. When $C = 10$, the estimate of σ is about 50% higher than σ_{JDH}, which results in similar predictive performance to the SVM (with σ_{JDH}), with fewer support vectors (Figure 5.6, *right*).

Why this would happen is unclear. When $C = 10$, there are fewer SVs, so one would expect their median separation to be larger, which would result in a higher estimate of σ. On the other hand, there are fewer training errors when $C = 10$ (since errors are being penalized more heavily) and thus more of the SVs are free ($\alpha_i < C$). For these experiments it appears that the net result is more free SVs when $C = 10$ and a correspondingly lower σ. When $C = 1$, the regularization is quite high, thus many points lie within the margin band. The few free SVs are spread out, which leads to a high estimate of σ. Thus the high explicit regularization (low C) has a knock-on effect (high σ) which increases the regularization further. The result is underfitting. Hence, it is likely that the optimal value of C, for a given data set and kernel, for LAIKA is higher than for the standard SVM (with σ_{JDH}). This is in agreement with the results of Chapter 4 (Tables 4.9 and 4.10).

When $C = 10$, the new version of LAIKA had the desired effect of adapting σ to only the free SVs. This resulted in fewer SVs being required to construct the decision boundary. The results are disappointing since the hope was that this would lead to improved accuracy, whereas the accuracy was unchanged. The quality of the SVM solution is not as sensitive to the choice of C as it is to the choice of σ. Hence LAIKA was developed as an attempt to automate the selection of σ. In doing so, it appears that LAIKA has become more sensitive to the value of C than is the SVM. Further experiments investigating the performance of LAIKA and the SVM for a larger number of values of C would be required to confirm or refute this.

In summary, the new version of LAIKA underfitted the data when the explicit regularization was high, owing to a knock-on effect. When the explicit regularization was lower, the new version of LAIKA gave the same predictive performance as the SVM. In the latter case, LAIKA reduced the number of SVs by 8% on average, at the expense of a near trebling in training time.

5.2.3 Summary of Heuristics

The heuristics have been applied to the classification of compounds as active or inactive on eleven competitive binding assays, using fourteen properties as descriptors. When using the SVM with a Gaussian kernel, with width σ_{JDH}, STAR led to SVM solutions with fewer support vectors, with no loss in predictive performance. Such solutions are quicker to evaluate, and potentially easier to interpret. An added bonus was that STAR was usually quicker to train than the SVM. The heuristic LAIKA did not improve the predictive performance and required excessive training time. It appeared to be more sensitive to the amount of regularization than the SVM. Using STAR to reduce the number of SVs may lead to a better results with LAIKA, but this was not possible owing to excessive training time.

5.3 Alternative Compound Representations

In Chapter 2 (Section 2.2), it was quoted from Kauvar et al. (1995) that 'the way in which a compound is represented ultimately limits the success of all subsequent procedures'. In particular, three possible representations of a compound were described. In the following, variations of these three representations are used to learn a decision function for classifying the competitive antagonists as active or inactive. In Section 5.3.1, the MACCS structural keys (Chapter 2, Section 2.2.2) are evaluated as a representation for these compounds. In Section 5.3.2, the reduced graph representation is used. This is a more sophisticated representation, similar to the fingerprint described in Chapter 2 (Section 2.2.3). Finally, in Section 5.3.3, a new type of affinity fingerprint (see Chapter 2, Section 2.2.4) is presented and evaluated.

5.3.1 Predicting from Structural Keys

To recap: a structural key is a bitmap in which each bit represents the presence or absence of a specific structural feature (James et al., 2000). Although structural keys were originally developed for database searching, they have proved to be useful for distinguishing active compounds from inactive ones (Brown and Martin, 1996). The structural key used here is the MACCS key (MACCS-II, 1994). The MACCS key of a compound is a 166-dimensional binary vector, where each bit indicates presence or absence of a particular two-dimensional structural feature. The results of Brown and Martin (1996, 1997) suggest that the MACCS key may be better for predicting biological activity than the properties used above in Sections 5.1 and 5.2. In their work, the MACCS keys

pK_I	Props		MACCS		t	p
	Test	s.e.	Test	s.e.		
1	0.22	0.01	0.19	0.01	1.24	0.11
2	0.27	0.01	0.21	0.01	3.54	0.00
3	0.26	0.01	0.21	0.01	2.76	0.00
4	0.25	0.01	0.21	0.01	2.38	0.01
5	0.33	0.01	0.29	0.01	2.26	0.01
6	0.37	0.01	0.31	0.01	2.92	0.00
7	0.30	0.01	0.28	0.01	1.20	0.11
8	0.27	0.01	0.19	0.01	4.47	0.00
9	0.31	0.01	0.19	0.01	6.92	0.00
10	0.24	0.01	0.19	0.01	2.45	0.01
11	0.24	0.01	0.21	0.01	1.71	0.04

Table 5.3: pK_i. Test error rates for the SVM when using the properties ('Props') or MACCS keys. The error on the MACCS keys is significantly lower on nine assays (t-test at 95%).

were shown to accurately predict the values of various physicochemical properties. The MACCS keys thus contain at least as much information about the compound as those physicochemical descriptors. Whether or not this extra information can be utilized for prediction depends on whether or not it is useful, and, if it is, whether or not a learning machine can take advantage of it. The SVM seems well suited to take advantage of this extra information since it is robust to high-dimensional data.

The kernel used for these experiments was the linear kernel:

$$K(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle = \sum_{p=1}^{166} x^p z^p = (1 - \text{Hamming}(\mathbf{x},\mathbf{z})), \tag{5.1}$$

where Hamming is the number of bits for which two bitmaps differ. This is a popular dissimilarity measure in chemoinformatics (see Chapter 2, Section 2.3.2). The resulting decision function is of the form:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{p=1}^{166} w^p x^p + b\right), \tag{5.2}$$

where w^p is the p^{th} component of the weight vector $\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i$. That is, once the SVM has been trained, a new compound is classified as being active if (and only if) the sum of the weights assigned to structural features that are present in that compound is greater than some threshold ($-b$). This is similar to the approach of Gillet et al. (1998), where a genetic algorithm was used to assign weights to the structural features.

The predictive performance of the SVM when trained using a linear kernel on MACCS keys compared to using a Gaussian kernel on the properties is illustrated in Figure 5.7. The performance is reported in terms of test errors, *RA*, and *AUROC* for $C = 1$. (The SVM did not converge within 10000 iterations for $C = 10$, no other values were tried.)

The test errors (Figure 5.7, *top left*) are represented in Table 5.3. On all 11 assays the test error when training on MACCS keys was decreased. The difference was significant on nine assays (t-test at 95%). *AUROC* (Figure 5.7, *bottom left*) was increased on all assays. *RA* (Figure 5.7, *top right*) was increased on all but assay 1. $\varepsilon^l_{\alpha\xi}$ correctly predicted that training on MACCS keys would lead to a lower test error than training on the physicochemical descriptors (Figure 5.7, *bottom right*). It should be noted that there is no reason to suppose that using $C = 1$ with a Gaussian kernel has any relation to using $C = 1$ for a linear kernel. For the Gaussian kernel the radius, R, of the data

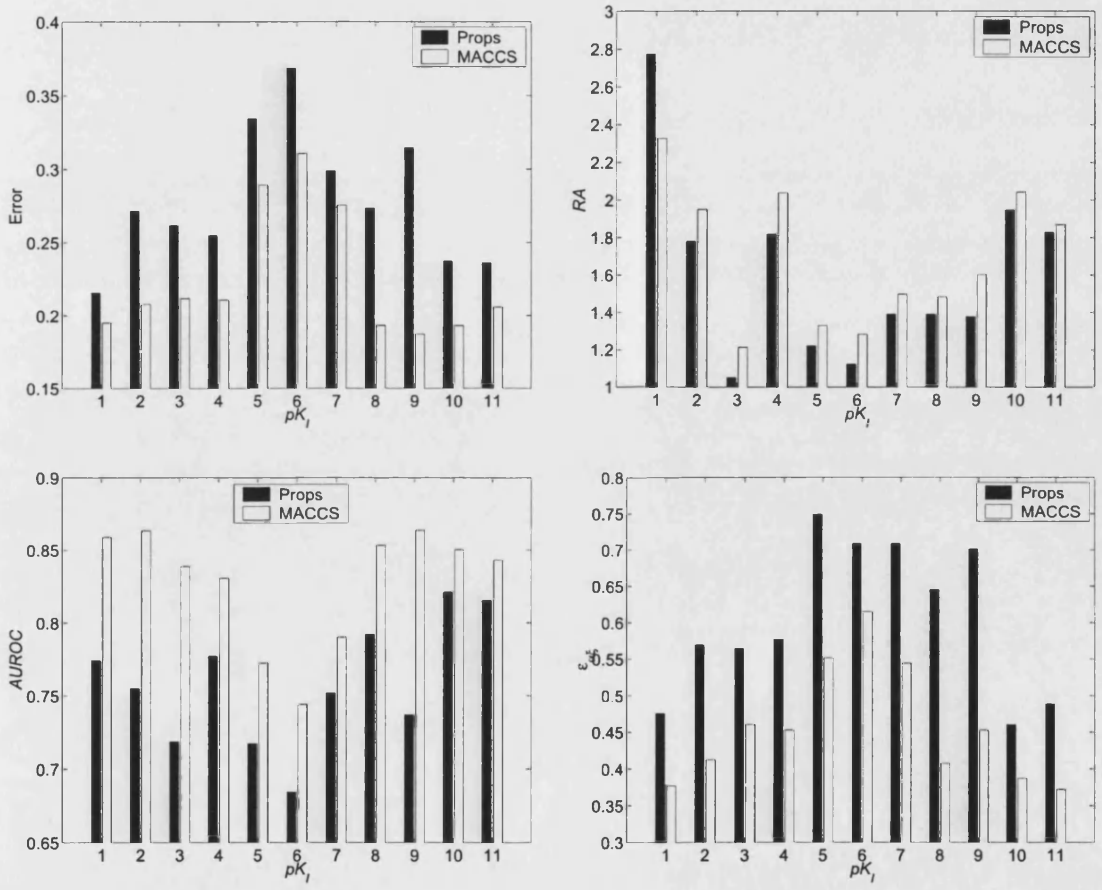

Figure 5.7: pK_i. Comparison of SVM performance when using the properties ('Props') or MACCS keys ($C = 1$): test error (*top left*), RA (*top right*), $AUROC$ (*bottom left*). Note that $\varepsilon^l_{\alpha\xi}$ (*bottom right*) correctly predicts that the structural keys will give better performance an all assays.

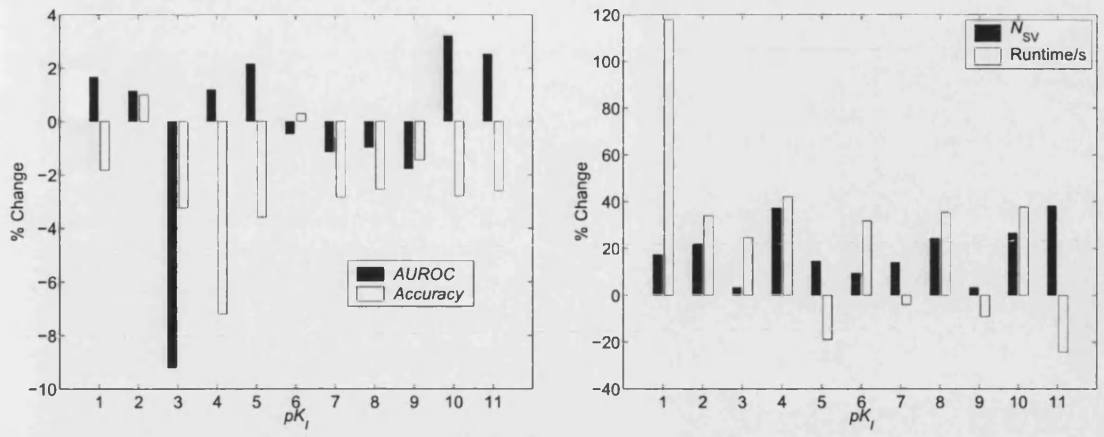

Figure 5.8: pK_i. When training on the MACCS keys, selecting the training data by clustering on fingerprints increases the *AUROC* on six assays, but this does not correspond to an increase in accuracy (*left*). The training time and number of SVs were increased (*right*).

is upper bounded by 1, whereas for a linear kernel on these data $R \approx 5$. If $C = R^{-2}$ (Cristianini and Shawe-Taylor, 2000), this suggests that better predictive performance could be attained with a smaller value of C when using the MACCS keys.

On average, the proportion of the training data that became SVs was 46%, compared to 78% when training on the properties. Thus the classification time would be reduced by about two-fifths. This gain was at the expense of a roughly seven-fold increase in training time. The solutions provided by training on structural keys are unlikely to be useful unless there are only a few informative training data and many unlabelled data. Such a situation may occur in the early stages of lead discovery.

5.3.1.1 Clustering Structural Keys

In Section 5.1 above, the training data were selected by clustering the compounds in the space of fingerprints. The SVM was then trained on the centroids, using the properties as descriptors. This resulted in a slightly denser model with a slightly higher *AUROC*. We would *a priori* expect the number of SVs to be fewer, since the centroids should be more representative of the data. When the SVM is trained on the centroids using the MACCS keys the results are as follows (again, for $C = 1$ only).

The *AUROC* was increased on six assays and decreased on five. However, these changes did not result in corresponding changes in accuracy (Figure 5.8, *left*). This implies that, for some of the assays, the ROC curve could be used to adjust the operating point to achieve a higher accuracy.

Training on the centroids usually led to an increased training time. The number of SVs was always increased, contrary to expectations. The percentage changes in training time and number of SVs when training on the centroids, compared to training on randomly selected data, are illustrated in Figure 5.8 (*right*).

In summary, when learning a predictor of activity from MACCS keys, clustering the data in the space of fingerprints did not lead to an improvement in predictive performance, and led to an increase in training time and classification time.

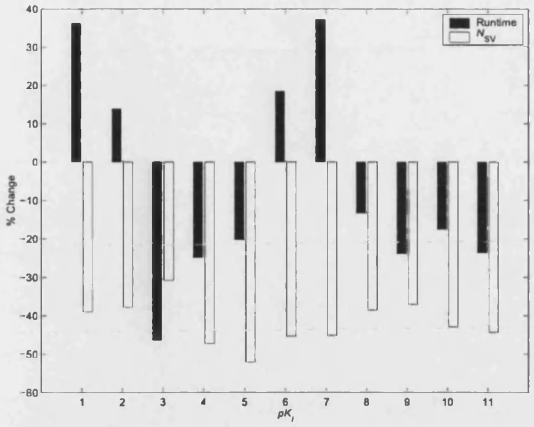

Figure 5.9: pK_i. When using structural keys as descriptors, STAR reduces the number of SVs by roughly one third to one half. The cost incurred is no more than a 40% increase in runtime.

5.3.1.2 STAR on Structural Keys

The predictive performance of STAR on these data was almost the same as that of the SVM on all three performance measures (with $C = 1$). The percentage difference in training time and number of SVs is illustrated in Figure 5.9.

STAR reduces the number of SVs by between a third and a half. The cost of this sparser solution is no more than an increase in runtime of about 40%. On average, the runtime of STAR is slightly lower than that of the SVM. The proportion of the training data that become SVs is about 20%–30%. This is a significant improvement on training the SVM on physicochemical descriptors. However, training with STAR on the structural keys requires three to 13 times more time than training the SVM on the physicochemical properties.

5.3.2 Predicting from Reduced Graphs

The results in the previous section showed that a representation directly encoding structural features, as opposed to derived physicochemical and structural properties, led to a sparser and more accurate solution. However, the training time was considerably increased. The MACCS keys were originally designed for searching for substructures in databases. There is no particular reason why the substructures encoded should be related to biological activity. Also, the topological relationship between the substructures is lost.

An alternative structural representation is the reduced graph (Gillet et al., 1991). A reduced graph is a simplification of the 2D chemical structure. As described in Gillet et al. (1999), the aim is to identify a level of simplification such that compounds with equivalent bioactivity are similar and those with different bioactivity are dissimilar. This is done by summarizing structural features that are likely to affect activity, whilst maintaining the topology between them. The reduced graph is then converted into a 2048-dimensional binary fingerprint.

The SVM was again trained with a linear kernel for $C \in \{1, 10\}$. The solutions were almost identical for the two values of C, results are reported for $C = 1$. The predictive performance when using reduced graphs was almost identical to that obtained when using MACCS keys on all three measures. However, the training error was always lower when using the reduced graphs, which may indicate that the SVM was overfitting in this case, in which case a lower value of C should lead to improved performance. (Also, the radius $R \gg 1$, which suggests $C \ll 1$.)

The main advantage in using the reduced graph representation, over using MACCS keys, for these compounds, was that the training time was reduced by a factor of between seven and 15. The number of SVs was 12%–28% higher than when using the structural keys. However, this is still 16%–41% lower than when using the properties.

It is noted by Gillet et al. (1999) that reduced graphs provide a complementary representation to standard 2D structural descriptors, and often identify active compounds that are missed when using the latter. Thus it may be possible to improve the predictive performance by combining the SVM models trained on the different feature sets.

The effect of STAR was similar to the case of training on MACCS keys. The predictive performance of STAR was almost identical to that of the SVM on all three measures. The training time was slightly (8%) longer. The number of SVs was reduced by 16%–42%, with 32%–46% of the training data becoming SVs.

Note that the reduced graph representation has some similarities to the document representation used by Joachims in the development of $\varepsilon^l_{\alpha\xi}$: it is high-dimensional, sparse, and binary. For such data, Joachims (2000) suggests that using $\rho = 1$ in the calculation of $\varepsilon^l_{\alpha\xi}$ (3.32) gives a tighter and more predictive bound, although this is not proved. On these data however, $\varepsilon^l_{\alpha\xi}$ selected the optimal value of C in $\{1, 10\}$ for only seven of the 11 assays ($p = 0.11$). (Note that the SVM converged to the global optimum for all experiments when using the reduced graphs, so $\varepsilon^l_{\alpha\xi}$ holds.)

In summary, the reduced graph representation led to the same gains in accuracy over using properties as did the MACCS keys. Training on the reduced graphs was quicker than training on the properties (which was much quicker than training on the MACCS keys). The reduced graphs also led to solutions with fewer SVs than did the properties (although more than did the MACCS keys). It is likely that the SVM was overfitting the data, and that a lower value of C would have led to improved performance. STAR reduced the number of SVs without deteriorating performance, at the expense of a slight increase in training time.

5.3.3 Predicting from Bioactivity

The aim of the work reported in this chapter has been to train an SVM to predict accurately which compounds will be active when screened against various targets. The approach has been to learn a relationship between some descriptors of the compound and its bioactivity. The descriptors used have been physicochemical and structural. The general principle underlying QSAR is that there is a relationship between these descriptors and affinity for a certain biological target. An alternative is to use biological affinities themselves as descriptors. This approach has been called *affinity fingerprinting* by Dixon and Villar (1998), see Chapter 2 (Section 2.2.4) for a description of their work.

A novel approach is taken here: the actual pK_i values themselves are taken as descriptors. That is, if the target is to predict whether or not a compound is active on assay 11 (say), then the descriptors are the pK_i values on assays 1–10. Thus, the binding affinities of a compound to a set of receptors are used to predict the affinity to another receptor. There are a number of advantages to this approach over the standard QSAR approach. Pharmaceuticals companies have a large amount of screening data. Thus when it desired to predict biological affinity for a new receptor, affinities for many other receptors are already known. Secondly, genome research has uncovered an enormous amount of receptor sequence data and research has begun to classify these receptors (Karchin et al., 2001). Decision rules relating affinities may aid this process. Receptors are categorized into families and superfamilies according to biological function. This existing domain knowledge may be drawn upon for selecting affinities as descriptors, thus guiding the process of feature selection,

Actual pK_I	SQ pK_I
<5.5	1
5.5–6.0	2
⋮	⋮
8.5–9.0	8
>9.0	9

Table 5.4: Rescaling of pK_I values to a semi-quantitative (SQ) ranking.

which remains a difficult problem in drug discovery (Dixon and Villar, 1998).

Note that, unlike *IC50* or percentage inhibition, pK_i values are directly comparable across screens since concentrations have been factored out. Thus the pK_i value of a ligand for a receptor can be considered a constant, i.e. an inherent 'property' of the ligand.

Since compounds are screened at only a limited range of concentrations, it is not always possible to report an exact pK_i value. The activity may be recorded as a number, or as less than or greater than a certain value. For these data, the concentrations used led to the data being left-censored at $pK_i < 5.5$ and right-censored at $pK_i > 9$. The pK_i values were thus converted to a semi-quantitative (SQ) scale, as in Table 5.4. A bin size of 0.5 was used, as the pK_i values are only accurate to approximately ± 0.2 units. The threshold for 'active', as determined by the screening group, was a pK_i of 6.0. Thus 'inactive' compounds have an SQ pK_i in $\{1, 2\}$ and active compounds have an SQ pK_i in $\{3, \ldots, 9\}$. It would be preferable to have bin sizes that better reflected the proportion of actives and inactives, but this was not possible owing to the precision and range of the assay.

An example thus comprises 10 integer-valued attributes in $\{1, \ldots, 9\}$ with a binary target ('active', or 'inactive'). Note that this is the most compact of the four compound representations used in this work. The SVM was trained on these data using a Gaussian kernel, with $C \in \{1, 10\}$, as for the properties. The test errors are compared to those of the SVM trained with the properties and with the reduced graphs in Figure 5.10 (*left*) and Table 5.5. (Recall that with the MACCS keys the SVM performance was almost identical to that that with the reduced graphs.) Results are shown for $C = 1$. This value was the optimal for the affinity fingerprint on all assays, although $\varepsilon^l_{\alpha\xi}$ selected it for only seven. Results are qualitatively the same for $C = 10$. On ten assays the

pK_I	RG		pK_i		t	p
	Test	s.e.	Test	s.e.		
1	0.19	0.01	0.20	0.01	-1.13	0.13
2	0.21	0.01	0.16	0.01	3.22	0.00
3	0.21	0.01	0.13	0.01	5.33	0.00
4	0.21	0.01	0.15	0.01	3.75	0.00
5	0.28	0.01	0.21	0.01	3.85	0.00
6	0.31	0.01	0.21	0.01	5.03	0.00
7	0.27	0.01	0.16	0.01	6.86	0.00
8	0.19	0.01	0.14	0.01	2.55	0.01
9	0.21	0.01	0.16	0.01	3.33	0.00
10	0.16	0.01	0.13	0.01	2.11	0.02
11	0.22	0.01	0.12	0.01	6.23	0.00

Table 5.5: pK_i. Test error rates for the SVM when using the reduced graphs ('RG') or affinity fingerprints ('pK_i'). The error on the affinity fingerprints is significantly lower on ten assays (t-test at 95%). Results are shown for $C = 1$.

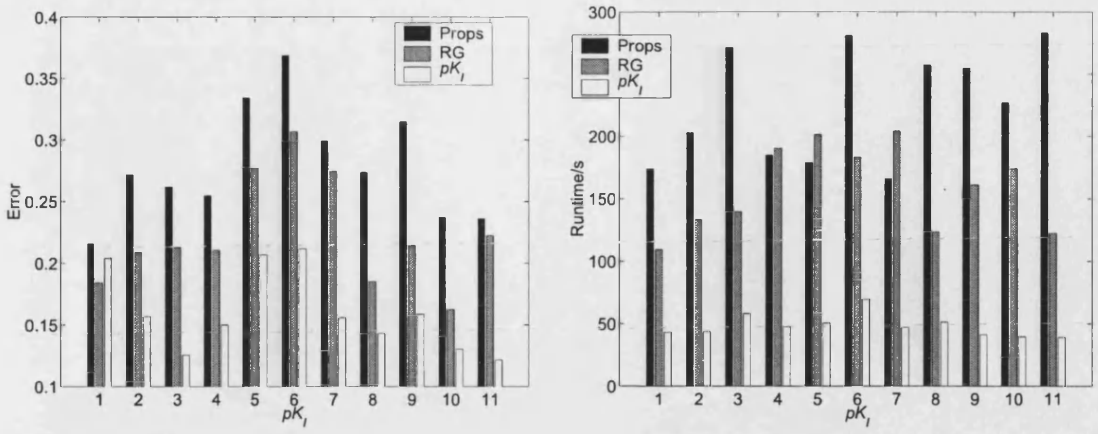

Figure 5.10: pK_i. The test error (*left*) when using affinity fingerprints ('pK_i') is lower than when using the properties ('Props') or reduced graphs ('RG'). The training time is also lower (*right*). (The test error for the MACCS keys was the same as for the reduced graphs; the training time was an order of magnitude longer.) Results are shown for $C = 1$.

affinity fingerprints led to significantly lower test errors than the other representations (t-test at 95%; for both values of C). On assay 1, no representation significantly outperformed the others. RA and $AUROC$ were also highest with the affinity fingerprints, except for assay 1. The gains in RA and $AUROC$ were relatively small (roughly 17% increase on average over using the properties) compared to the gain in accuracy (39%).

The affinity fingerprints led to solutions with more SVs than did the reduced graphs when $C = 1$ and fewer SVs when $C = 10$ (although this latter was suboptimal for both representations). The number of SVs for both of these representations was in general lower than when training on the properties and higher than when training on the MACCS keys.

The training time was much lower when using the affinity fingerprints than when using any of the other three representations (Figure 5.10, *right*).

In summary, binding affinities provide a highly compact and informative representation of a compound. This facilitates rapid training of the SVM for prediction of other binding affinities. The solutions are sparse and the most accurate developed in this work.

5.3.3.1 STAR on Affinity Fingerprints

The predictive performance of STAR, on all three measures, was fractionally poorer than the SVM, for both C in $\{1, 10\}$. $AUROC$ was one percentage point lower on nine assays when $C = 1$ and on four assays when $C = 10$, and equal (to the nearest percentage point) on the remaining cases. The RA of STAR was within four percentage points of that of the SVM, and was on average unchanged. The deterioration was slight but consistent.

The number of SVs was reduced on average by 20% at $C = 1$ and by 8% at $C = 10$ (Figure 5.11). The corresponding training times were on average 32% and 37% longer. Thus, if the better value of C could be selected, STAR would lead to a reduction in prediction time of one fifth, at the expense of an increase in training time of one third. If there are many more examples on which predictions are desired than for which activities are known, then STAR offers a reduction in overall computational time. This is usually the case when building decision rules in drug discovery.

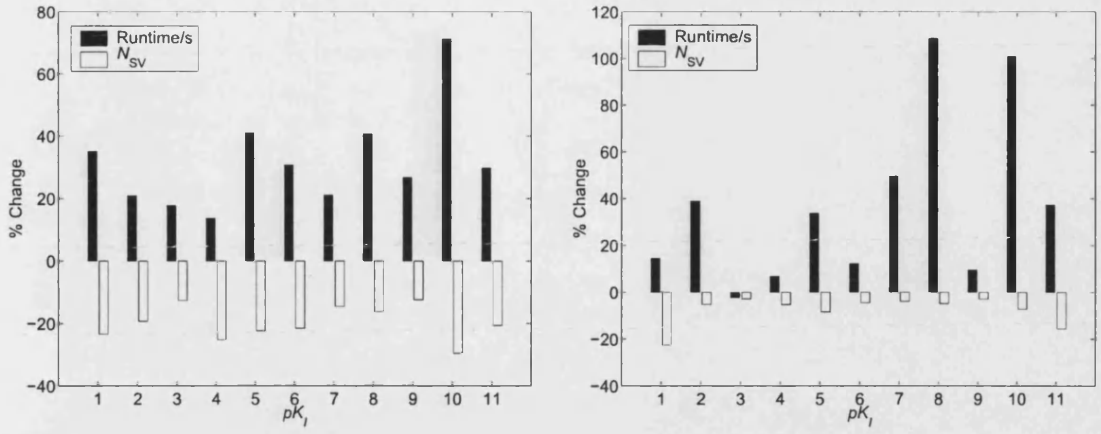

Figure 5.11: pK_i. STAR reduces the number of SVs (N_{SV}) at the expense of longer runtime. The relative gain compared to the increased time is larger for $C = 1$ (*left*) than for $C = 10$ (*right*)

5.3.3.2 LAIKA on Affinity Fingerprints

Since the number of iterations required to train the SVM on these data when $C = 1$ was low (< 300), LAIKA had no effect on most of the assays. LAIKA is only likely to have an effect if the update period, h_a, is much smaller than the number of iterations. Since this is not known in advance h_a was set at 100 for all experiments reported here. If the training is completed rapidly, one possibility is to estimate σ from the final set of SVs and retrain.

On the two assays where LAIKA had an effect when $C = 1$, the estimate $\hat{\sigma}$ was approximately $3\sigma_{\mathrm{JDH}}$. This resulted in a slight increase in $AUROC$ and a substantial decrease in the number of SVs, at the expense of two to three times longer training time. However, the test error increased (by one percentage point) on these two assays. This is again indicative of the operating point b being suboptimally set for a heuristic.

When $C = 10$ the number of iterations to convergence of the SVM was 300–600, and LAIKA had some effect on the SVM. The predictive performance was consistenly but not significantly better than that of the SVM. The number of SVs was reduced, roughly in proportion to the increase in σ (Figure 5.12). Note that, as before (Section 5.2.2), σ is lower when $C = 10$ than when $C = 1$ (for those assays where LAIKA had an effect). The cost of this is up to six times longer training time, which is moderately correlated ($\rho = 0.68$) with the reduction in the number of SVs. It seems unlikely that these results would be of any use in practice.

5.3.3.3 STAR+LAIKA on Affinity Fingerprints

When $C = 1$, the $AUROC$ of STAR was slightly decreased on eight assays. The $AUROC$ of LAIKA was increased slightly on two, and the solution remained the same on the others. STAR decreased the number of SVs, which should increase their separation, and hence the estimate of σ. One might expect that LAIKA could recover the performance of STAR to that of the SVM (or better) whilst maintaining the sparsity conferred by STAR. This is what happened: the $AUROC$ of the combination was slightly increased in eight cases, decreased on one (assay 1) and was unchanged on the other three. (Comparisons are to within one percentage point.) The number of SVs was on average unchanged from that of STAR. On only one assay (again, assay 1) was the number of SVs of STAR+LAIKA higher than that of the SVM.

The RA of STAR+LAIKA was correspondingly higher than the SVM or unchanged. The test

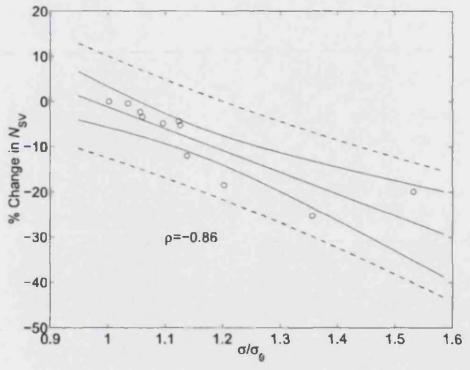

Figure 5.12: pK_i. When $C = 10$ LAIKA increases the estimate of σ over its starting value (σ_0), leading to a reduction in the number of SVs (N_{SV})

error was not always reduced (and was sometimes increased) when *AUROC* was increased. This is again indicative of a suboptimal operating point.

As expected, the sparsity induced by STAR led to increased separation of SVs and an increased LAIKA estimate of σ. The number of SVs and the *AUROC* were weakly correlated ($|\rho| \approx 0.5$) with the increase in σ.

The cost of this fractional increase in predictive performance, and reduction in number of SVs by a fifth on average, was nearly a doubling in training time. On the two assays (3 and 6) where LAIKA affected the solution, the combination was quicker than LAIKA. However, the combination was always slower than STAR. Thus on these data it would be preferable to use STAR+LAIKA over STAR alone only if the decrease in performance of STAR were unacceptable.

When $C = 10$ the conclusions are similar. The *AUROC* was fractionally increased above that of the SVM, and STAR+LAIKA recovered the performance of the SVM on those assays where STAR deteriorated it. The reduction in the number of SVs was lower (15%). However, the training time was increased dramatically, by a factor of 27 over the SVM. (The largest increase in training time for the heuristics individually was a by factor of six.)

In summary, STAR+LAIKA provided the benefits of both heuristics, viz. reduced number of SVs with (slightly) improved performance. The effect was more pronounced when $C = 1$, which required less than twice the training time of the SVM. With a higher value of C the minor gains were rendered Pyrrhic by the extensive training time.

5.3.4 Summary of Alternative Representations

All three alternative compound representations — MACCS keys, reduced graphs (RG) and binding affinities (pK_i) — offered improvements over the physicochemical and one-dimensional structural descriptors ('properties'). The predictive performance was best for pK_i. RG and MACCS gave the same predictive performance and were better than properties. MACCS gave the sparsest solutions, but at the expense of the longest training time. Thus MACCS keys would be preferable for learning a rough and ready filter from a few screened compounds. This could be used to select compounds for screening from a large (possibly virtual) library. Since training on MACCS keys is time consuming, a procedure to select informative training data is desired. Using Ward's clustering to select the training data did not lead to improved performance. The characteristics of SVM training and solutions for the four compound representations are summarized in Table 5.6.

Props	MACCS	RG	Affinities
		Accuracy	
Worst; SVM is overfitting so could be improved	Good	Good	Best
		Training time	
Slow	Slowest	Fast	Fastest
		Classification Speed	
Slowest	Fastest	Fast	Fast
		Interpretability	
Properties are interpretable but solutions were dense	Common structural features amongst the few SVs can be identified	Very poor	Open question; could provide a similarity metric for receptors

Table 5.6: pK_i. Summary of SVM training and solutions for the four compound representations.

RG and pK_i gave solutions sparser than did properties and were quicker to train. pK_i led to the shortest training times. Thus pK_i would be preferable when high accuracy is needed and there are large number of screened compounds from which to learn a decision rule.

These results suggest the following procedure.

1. Randomly select a small set of compounds, from a compound library.
2. Screen these *in vitro*, i.e. a low-throughput screen.
3. Using MACCS keys, learn a sparse classifier.
4. Predict the activities of the compounds in the library and rank them in order of predicted activity, e.g. as 'active', 'moderately active', 'inactive'.
5. Screen the 'active' compounds *in vitro*, i.e. a high-throughput screen.
6. Using other pK_i values where available, or RG otherwise (or a combination), learn a more accurate classifier.
7. Use this a second, more accurate, virtual screen for the 'moderately active' compounds. The compact representation may also suggest relationships between the targets.

How many compounds are selected for the *in vitro* assays will depend on time and cost resources. Use of the classifiers should make the process of finding actives in the large compound library more efficient.

STAR always led to sparser solutions than did the standard SVM. The performance was the same as the SVM for RG and MACCS, and deteriorated slightly on pK_i. The training time was never increased by more than 40%. In the above schema, STAR would be best employed when training with MACCS at Step 2, in order to reduce classification time.

LAIKA slightly improved the performance of the SVM trained on pK_i, and led to sparser solutions. However, the training time was increased considerably. LAIKA would not be useful in the above schema.

5.4 Summary

In Chapter 3, the pK_i data were described. An SVM was used to predict whether or not a compound was active against various targets. The compound description used consisted of fourteen physicochemical and one-dimensional structural descriptors. The SVM solutions had a high proportion ($> 50\%$) of support vectors and had poor predictive performance. This prompted the development of three heuristics, as described in Chapter 4. Three approaches to improving the SVMs performance have been presented in this chapter. These were: two general purpose data mining methods; the application of the heuristics; and, the use of alternative compound representations. These approaches are not mutually exclusive. The main contributions presented in this chapter, and some ideas for future work, are summarized below.

Selecting Training Data Screening compounds against a target costs money and takes time. Also, the training time of the SVM is $O(l^2)$–$O(l^3)$. Therefore, it is desirable to select a subset of the available compounds to physically screen in order to train an SVM. Ward's clustering, a fast and popular algorithm in chemoinformatics, was applied to this task. The clustering was performed in the space of Daylight fingerprints using the Tanimoto similarity measure.

- The clustering revealed that the fourteen properties were insufficient to fully discriminate the compounds.
- On both properties and MACCS, the test accuracy was in general poorer. However, the $AUROC$ increased. This suggests that the operating point was incorrectly set. This was probably because the class priors were different in the training and test sets.
- Prior knowledge of the densities could be used to adjust the regularization constants for the two classes. Failing that, the ROC curve could be used *post hoc* to correct the operating point.
- Overall, whether minimizing expected error or expected risk, randomly selecting training data was preferable. In this case, the number of SVs and training time were slightly lower, and RA was slightly higher.

Minimizing Weighted Loss When using a classifier for screening set selection, one aim is to increase the number of active compounds found. This can be quantified by the relative advantage (RA), a measure similar to the global enhancement over random selection of compounds. A specific risk functional was minimized as a proxy for maximizing RA.

- Although it is not guaranteed, $AUROC$ was almost the same whether minimizing error or risk.
- When minimizing the respective loss, risk was more highly correlated with $1/RA$ (and $AUROC$) than was error.
- However, minimizing risk led to a decrease in RA, contrary to expectations. The risk functional is a first order approximation. A refined version is suggested in Chapter 6 (Section 6.2.3.3).
- The number of SVs was slightly higher. This may be because the hyperplane was pushed towards the over-represented class. Such an hyperplane would make less mistakes on the under-represented class, since it is further away. If the hyperplane lies in a region where examples are denser, then, intuitively, there will be more SVs.

HERMES This heuristic stopping criterion was used for all of the experiments reported in this chapter. The experiments with STAR and LAIKA would not have been possible without some sort of heuristic stopping criterion. A comparison of the solutions obtained with those obtained at the global optimum of the QP would demonstrate any advantages of HERMES.

STAR This heuristic for reducing the number of SVs was evaluated with all four compound representations. Overall, the behaviour was as expected: the number of SVs was reduced with little or no loss in predictive performance.

- The number of SVs was reduced on all assays for all representations.
- The predictive performance (w.r.t. all three measures) was almost the same as that of the SVM. It was slightly worse than the SVM on pK_i.
- The training time was usually reduced on pK_i and MACCS. It was slightly longer on RG and about a third longer on pK_i.

LAIKA This heuristic for tuning the width of a Gaussian kernel was evaluated with the properties and pK_i representations. Overall, the performance, in terms of both time- and space-complexity, was disappointing.

- On properties, the predictive performance deteriorated w.r.t. all three measures.
- On pK_i, the predictive performance was fractionally better than the SVM. The operating point may have been set suboptimally on some assays.
- The number of SVs was slightly reduced for both representations.
- The number of iterations and the training time were considerably increased.
- It is postulated that the explicit regularization should be lower. However, this is achieved by increasing C, which would increase the already lengthy training time.

MACCS Keys This representation is a 166-dimensional bit string encoding presence of predefined substructures within a molecule.

- The predictive performance was better than that of the properties on all three measures.
- $\varepsilon^l_{\alpha\xi}$ was within a factor of two of the test error, and correctly predicted that MACCS keys would give lower error than did properties.
- The number of SVs was reduced, resulting in a 40% reduction in classification time.
- These gains were at the expense of a seven-fold increase in training time.
- The SVM was overfitting the training data. It is likely that a lower value of C would lead to lower error (and shorter training time).

Reduced Graphs This representation is a 2048-dimensional bit string that aims to encode structural features relevant to binding, whilst maintaining their topology.

- The predictive performance was almost the same as that of the MACCS keys (i.e. better than that of properties).
- The training time was much faster than that of MACCS and less than that of properties.
- The number of SVs, and hence classification time, was more than that of MACCS, but still lower than when using properties.

- As for the MACCS keys, the SVM was overfitting. It is likely that a lower value of C would give better performance.

Affinities For a given receptor, this representation comprises the binding affinity of a compound to the other ten receptors. This is a novel representation, developed with Steven Barrett at GlaxoSmithKline.

- On ten of the assays, this gave the best performance on all three measures.
- The number of SVs was about the same as the reduced graphs, i.e. more than when using MACCS and less than when using properties.
- The training time was less than half of the next quickest, viz. reduced graphs.

HERMES and STAR are broadly successful and, when applied to the problems in drug discovery studied here, achieve the objectives for which they were developed. LAIKA, however, was not successful when applied to these problems. The use of alternative compound representations, such as the MACCS structural keys and reduced graphs which are more sophisticated than chemical property data, was beneficial. The novel use of affinity data which potentially provides a more direct source of information than the other representations, was even more beneficial.

The principal message of this chapter has been that there are numerous ways to tackle a data mining problem. Which method is preferable depends on the practical situation. Do we want a solution rapidly? How many data are there? How accurate does it need to be? At what stage in the drug discovery process are the predictions being made? The heuristics, compound representations, and data mining methods presented here may help or hinder the data miner depending on the answers to these questions.

Chapter 6

Critical Assessment, Further Work, and Conclusions

> *The real question is not whether machines think but whether men do.*
>
> B.F. SKINNER

The work presented in this thesis has been primarily concerned with the computer science field of supervised learning. Specifically, the classification of objects into one of two predetermined classes, or the ranking of objects in terms of some real-valued variable. Broadly, the field of supervised learning poses three questions: What is it theoretically possible to learn? How is this achieved? What actually happens when we apply learning algorithms to practical problems? The first question is not addressed in this work. Statistical learning theory provides one answer to this question. This inspired the development of the support vector machine (SVM) in answer to the second question. The motivation behind this work was to answer the third question, with respect to the learning problems that arise in the early stages of drug discovery.

The early stages of drug discovery have been described, as have some previous applications of machine learning. The aim of using such techniques is to improve the efficiency of drug discovery by saving time and money. It is also hoped that these automated techniques may contribute to pharmacological knowledge. The SVM was a promising new technique that had not previously been applied to predicting the biological activity of compounds. A common approach to predicting activity is quantitative structure-activity relationship analysis (QSAR). This attempts to relate physicochemical and structural descriptors to a real-valued activity. The obvious[1] next step was to apply the SVM to a public domain QSAR problem and compare it to other techniques (Burbidge et al., 2000, 2001a). The SVM proved to be the favourable algorithm. However, this was a small (55 compounds), well-characterized prepared data set. SmithKline Beecham (now GlaxoSmithKline) were interested in predicting the activity of compounds from data generated at the earliest stages of drug discovery. Standard QSAR was not designed with such data in mind. The SVM seemed well-suited to the task.

Two problems were chosen. The first ('HTS') was to predict the activity of compounds from data generated from a single high-throughput screen. The second ('pK_i') was to predict the activity of compounds from data derived from several screens. The SVM was applied to each of these problems and was found to suffer from some problems not typically reported in the literature.

[1] Obvious to M. Trotter, that is.

	C5.0	Neural Networks	SVM
Accuracy	Has proved poor here and in previous work at GSK	Poor without extensive model selection	Not as good as expected; LAIKA gave little gain
Unbalanced Data	Can be tricky (e.g. on HTS, not presented here)	Poor on HTS	Easy to implement; risk bounds become unreliable
Classification Speed	Fast	Reasonably fast	Slow owing to dense solutions; STAR improves
Interpretability	Easy	Very difficult	Open question
Robustness	Poor	Unreliable	Dependent on parameter selection
High-dimensional	Not addressed	Poor without extensive model selection	Good
Training time and Model Selection	Very fast	Tediously slow, especially with model selection	Reasonably fast, easy model selection; HERMES improves
Memory Requirements	Low	Low for online learners used here	Can be large although online algorithms available

Table 6.1: Method-to-criterion. A brief summary of my learnings for three algorithms, when applied to pharmaceutical data analysis.

Three heuristics were developed in an attempt to overcome these deficiencies.

LAIKA The purpose of LAIKA (Burbidge, 2002a) is to automatically tune one of the few free parameters of the SVM. The aims were (in order of importance): to improve accuracy over using a predetermined heuristic; to reduce model selection time; and, to reduce the solution complexity and thus reduce classification time.

STAR The purpose of STAR (Burbidge et al., 2001b) was to automatically remove noisy data points. The aims were (in order of importance): to reduce solution complexity and hence classification time; and, to reduce training time, without loss in performance.

HERMES The purpose of HERMES (Burbidge, 2002b) was to provide an alternative stopping criterion for SVM training. The aims were: to reduce training time without loss of performance; and, to provide a stopping criterion when the other heuristics are used.

These heuristics were evaluated on public domain data sets and shown to meet their various aims to various degrees. They were then applied to the SVM classification of the pK_i data.

In Chapter 1, Section 1.2, some desirable properties of a machine learning algorithm in the context of drug discovery were outlined. In Table 6.1, I briefly summarize my learnings with respect to these criteria for C5.0, neural networks and the SVM. The observations apply only to pharmaceutical data analysis of the kind presented in this thesis. The criteria 'Chemical Sense' and 'Missingness' have not been included as it is not possible to infer anything regarding these from this work.

6.1 Critical Assessment

The contributions presented in this thesis are of two types: heuristics for the SVM: HERMES, LAIKA, and STAR; and, analysis of drug discovery data sets: QSAR, HTS, and pK_i. In the

following, the work pertaining to each of these six sub-topics is critically assessed and the main contributions, as italicized in the Synopsis (Chapter 1, Section 1.5), are restated.

6.1.1 General Criticisms

Before considering the contributions presented in this thesis, it is worthwhile to consider some generalities of data mining and SVMs. Various points have been glossed over during the analysis, and numerous claims of others have been left unsubstantiated w.r.t. drug discovery data.

- In providing estimates of performance, e.g. the error rate, the various algorithms have been evaluated w.r.t. a test set disjoint from the training set. For a given data set, say the standard SVM on the first of the 11 pK_i assays with the MACCS representation, the error on the test set is an unbiassed estimate of the true error. It is common practice, as here, to evaluate several heuristics, with various representations, and choose that combination with the lowest test error. We can reasonably expect this combination, call it Learner A, to perform well on further unseen data. However, the lowest test error of a collection of learners is not an unbiassed estimator of the true error of Learner A. (Consider 10 random classifiers, over repeated trials, the lowest of the 10 test errors will always be < 0.5 but the true error of the particular classifier achieving that minimum each time is 0.5.) A sounder approach would have been to ensure that there was a further set of data, from the same data generating process, on which to evaluate the algorithms. The practicalities of this for the pharmaceutical data sets are dealt with in the relevant sections below.

- It is claimed that the support vectors (SVs) are 'in some sense the most important patterns in the data' (Burges, 1998). The SVs found in the analysis of the drug discovery data sets have not been analysed by chemists or pharmacologists.

- SVM errors have been penalized linearly throughout this work (i.e. $k = 1$ in Eq. 3.15). Drug discovery data are noisy, so this seems preferable to quadratically penalizing errors (i.e. $k = 2$). The latter has not been evaluated. If it led to comparable performance, then it may be preferable since the VC bound (3.28) could then be used for parameter selection.

- The 'unbiased' hyperplane (the solution to the QP (3.22) has been used throughout this work. This gives the same performance as the 'biased' hyperplane (the solution to the QP (3.15) on public domain data sets. The difference has not been investigated on the pharmaceutical data sets.

- The estimator $\varepsilon^l_{\alpha\xi}$ was in general not reliable for parameter selection. The evaluation in Chapter 3 (Section 3.1.5.3) was not very thorough. Further, the suggestion that solutions where the SVM predicts all examples to be in the same class should be discarded was not implemented.

- The area under the ROC curve ($AUROC$) has been reported as a performance measure since misclassification costs are not known. However, it may be possible to obtain estimates of costs at different stages. It may be that only the behaviour for a subsection of the ROC curve is of interest.

- The relative advantage (RA) has been reported as a performance measure. As noted, it is not very meaningful when the true positive ratio is very small, but this was not checked during the analysis of the results. GE is a more reliable measure (and is used at GSK), but

it requires a ranking of compounds. (Note that $RA = GE$ when the operating point is such that the true positive rate is 0.5.) It would have been preferable to calculate GE for all of the reported results.

- The risk functional chosen to maximize RA did not give as high RA as minimizing the error rate, at least on the pK_i data. There was a high correlation between risk and RA^{-1} for the public domain and pK_i data sets. There are probably better ways, possibly iterative, to maximize RA.

- All features were scaled to lie in the range $[-1, +1]$. This is common practice for the SVM (Lee and Lin, 2000; Hsu and Lin, 2002). Theoretically, this should make no difference since there is a global optimum. What the effect is in practice remains unassessed. A common procedure in statistical pattern recognition is to scale the features to have mean 0 and variance 1 (Ripley, 1996).

- The Adult data sets were pre-processed as in Platt (2000). Joachims (2000) suggests that this is a good idea. The SVM was not evaluated on the original data.

- It is claimed that the parameters of the optimizer do not affect the quality of the solution (Chapter 3). Theoretically this is true. In practice, the results can differ slightly, particularly for the larger Adult data sets (Chapter 4, Section 4.1.2).

- The Adult data sets are subsets of a much larger data set. Thus there it is possible to evaluate the learned rules on a large unseen test set to obtain better comparisons of performance.

- The Matlab implementations of the SVM and the heuristics run about 100 times more slowly than an equivalent implementation in C or C++. This seriously limited the number of experiments that could be carried out. In particular, it would have been preferable to carry out extensive characterization of the heuristics on artifical data sets of different sizes, dimensionalities and signal-noise ratios.

Overall, the various analyses have highlighted the benefits of deciding in advance what exactly it is that the data mining exercise is supposed to achieve. This should then motivate the choice of performance measure and learning algorithm. Exploratory data analysis is also useful, but it should precede, not replace, a systematic investigation.

6.1.2 QSAR

The aim of this analysis was to predict the inhibition of dihydrofolate reductase by pyrimidines. The inhibition was reported as $\log(1/K_i)$ ($=pK_i$, but the symbol 'pK_i' is used in this thesis as an abbreviation for the data sets analysed in Chapter 5). This problem is typical of those encountered in standard QSAR analysis (Eriksson and Johansson, 1996; Hansch and Leo, 1995): there are a small number (55) of compounds whose real-valued acitivity we wish to predict. King et al. (1992) reformulated this as relational problem in order that inductive logic programming (ILP) could be applied. The aim is to learn the relationship great(). The advantages of this are that ILP can provide a few human-readable rules with high accuracy. They found that the learned rules could successfully rank compounds in the training set, but not compounds in the test set. The relational problem suffers from some disadvantages:

- The relation great() is not defined for drugs of equal activity.

- The number of examples is increased from l to $O(l^2)$, which is likely to increase training time.
- Converting relations to a ranking can lead to the Condorcet (Escher staircase) paradox.

Also, when the work of Burbidge et al. (2000) was presented, doubts were raised about the suitability of the compound representation used by King et al. (1992).

The relational problem can also be solved by a classifier such as the SVM. In hindsight, this seems rather pointless.

- SVM regression on 55 compounds with 27 features would be much quicker than SVM classification on 2800 comparisons with 54 features.
- The solutions would also be more readable.
- As a classification problem the advantages of learning a relation have been lost.
- It is possible for the classifier to predict that $\text{great}(n, m) = +1$ and $\text{great}(m, n) = +1$.
- Defining a threshold for 'active', as for the pK_i and HTS data, would have been a better way to frame this as a classification problem.
- The error on each test set was around 11%, which is the number of comparisons $\text{great}(n, m)$ for which neither d_n or d_m appeared in any training comparison. The question is: Have we simply learnt a ranking of training compounds and not of test compounds?

Overall, casting the relational problem as a classification one was ill-advised and did not lead to any new insight from the point of view of QSAR analysis.

From the point of view of machine learning the following points were highlighted by the analysis.

- Fast techniques such as a decision tree and automatically tuned neural nets did not perform well.
- Extensive time and human effort was required to obtain good performance from the neural network.
- The SVM was fast and easy to tune, and gave optimal performance of the methods assessed.

Some criticisms of the SVM methodology used are as follows.

- The estimator $\varepsilon^l_{\alpha\xi}$ was used to select the parameters C and σ. This gave good performance for these data, but results presented throughout the thesis suggest that $\varepsilon^l_{\alpha\xi}$ is not reliable. It would have been preferable to use a validation set for parameter selection, as for the other techniques.
- Nine parameter settings were tried, based on fixed heuristics. There is no evidence that these are optimal for these data. A more exhaustive search, e.g. by LOOMS, would show whether the heuristics were suitable.

The training errors were lower than the test errors, which suggests that the SVM was overfitting and that the above criticisms are valid. However, the training and test sets were not i.i.d. so it is not possible to assert this. Further analysis is not warranted, since the problem should be reformulated.

As a final, general point, this data set also highlighted the advantages of formulating the problem sensibly in the first place, as opposed to diving blindly in with the latest piece of technology.

6.1.3 HTS

The aim of this analysis was to rank unscreened compounds from the compound library for high-throughput screening (HTS). Screening data were made available by SmithKline Beecham (now GlaxoSmithKline, GSK) for five classes of receptor. Only the data from the class 'a' screens were used in the analysis. There were data for eight specific receptors of class 'a'. Only seven of these were used as there were insufficient data for receptor a4. For the purposes of evaluating the learned decision rules, a quarter of the screened compounds were treated as unlabelled. When learning the decision rules, the remaining three quarters were partitioned into training and test sets of equal sizes. The results reported in this thesis are the results on these test sets. Owing to time constraints, the decision rules were not evaluated on the third set of 'unlabelled' compounds (see Section 6.1.1).

Several feature sets were provided for the compounds. Three of these, fs01–fs03, were used in this analysis. It would have been preferable to have used all of the feature sets, either singly or in combination, and use the test sets for feature selection. The 'unscreened' compounds could then have been used for a validation of the entire procedure. Further, once a procedure for screening set selection had been developed, the data for the other four receptor classes could have been used for further validation.

A further aim, initially, was to identify which features were useful for prediction, for a given receptor, a given receptor class, and for all receptors in general. Initial attempts (not reported here) proved this to be a difficult problem. The analyses in Chapters 2 and 3 showed that no one feature set was optimal for receptor class 'a'.

Overall, the analysis of the HTS data, though far from exhaustive, was carried out well.

- Exploratory data analysis, such as visualizing the distribution of the features, did not reveal anything particularly unusual.
- Techniques already used in drug discovery for classification and regression were evaluated. The linear techniques performed well.
- The SVM for classification did not perform well, even with extensive model selection.
- When the low proportion of hits was taken into account the SVM performed well, without extensive model selection.

These initial results suggest that the SVM, adjusted for the low hit rate, could be a more useful tool for screening set selection than the linear and neural net techniques currently used in QSAR. As it was set out to show, techniques used for QSAR are not best-suited for the analysis of HTS data.

The following criticisms could be made of the analysis.

- Only automatically tuned neural networks were used. The analysis of the QSAR data showed these to have poorer performance than manually tuned networks. Since there were 21 problems with thousands of examples each, manually tuning the neural networks would have required much longer computational time and human effort.
- The SVM was shown to benefit from assigning a higher misclassification cost to false negatives. This may also lead to better performance for the other techniques.
- The results have not been assessed by the screening group at GSK. It is not known how useful they are.

- Domain knowledge was not incorporated. This was very difficult to obtain.
- Although the global enhancements (GE) are better than random, i.e. $GE > 1$, it is not known whether these results are statistically significant. The $A50$ follows a negative binomial distribution, so it should be possible to develop a significance test.
- The techniques of clustering and similarity-based prediction have been described for screening set selection. The SVM, and other techniques have not been compared to these.

There is plenty of scope for further work.

Analysis of these data has highlighted that linear techniques can do better than their more sophisticated rivals, and that even the most recent advances in machine learning can fail without consideration of the characteristics of the data.

6.1.4 pK_i

The aims of this analysis were exploratory. GSK provided screening data for over a thousand compounds against 11 receptors. One broad aim was to assess the usefulness of various compound representations for learning from such data. A new representation was also proposed, whereby the aim was to relate affinities for different receptors. This would aid the process of characterizing the different receptor types and subtypes. Analysis of the human genome has led to the discovery of many new receptors (Howard et al., 2001); any relationships between them could facilitate their categorization. Compact relationships are desired, since these are more easily interpretable in terms of biological and chemical sense. From a machine learning viewpoint then, the aims were to learn a compact predictor of activity for all of the receptors. Since the application remains open, the predictive performance was assessed w.r.t. a variety of measures. The error rate records the number of misclassified compounds, an obvious measure of accuracy (at least when the classes are not too unbalanced). The ROC curve gives a measure of the performance for a range of costs. This is more useful in a practical scenario. For example, the classifier may be used as a filter that flags unscreened compounds as 'probably active'. The maximum allowed false positive rate (f.p.r.) will depend on screening costs, the ROC curve gives the hit rate for a specified f.p.r. The ROC curve on the training data can also be used to alter the threshold above which compounds are deemed active. The f.p.r. and hit rate for all costs are summarized as the $AUROC$. If the classifier is used for screening set selection, then the global enhancement (GE) over random selection is useful. A similar measure, the relative advantage (RA) is reported here. Note that the maximum this can attain is $1/\pi_{+1}$, which for these data is a modest 1.4–3.6 gain over random selection of compounds. However, if the classifiers were used to select compounds from a company's library then π_{+1} would be lower and the potential gains much higher. In this case, care would have to be taken to adjust the misclassification costs to take account of the different hit rates, and to ensure that the training data were representative of the library compounds.

Overall, the analysis of these data led to a number of useful conclusions regarding: the feasibility of such learning; the discriminative power of the feature sets; the performance of the SVM; and, the usefulness of the heuristics; as summarized in Chapter 5. Here, I outline some aspects of the analysis which, in hindsight, could have been better carried out.

- The compounds were deemed active if $pK_i > 6$, if this is varied, then a different classifer results, with a different weight vector $\mathbf{w}$. If the rankings obtained for different activity thresholds vary, then doubt would be cast on the validity of producing a ranking in such a way.

- If a ranking is desired, e.g. if the application is screening set selection, then a regression technique may be preferable. Particularly as the signal-noise ratio was high (about 30). The raw data were left- and right-censored, and the preprocessed data ordinal, so the regression approach would not be straightforward.
- It would have been preferable to identify the best representation for the SVM first. The other techniques and heuristics could have then been evaluated against the SVM on just this representation, thus saving time and effort. As this work was primarily exploratory, the representations were suggested and calculated by Steven Barrett of GSK during the research in an attempt to improve the performance.
- A similar point is that all experiments were carried out using the randomly selected training data and the centroid training data simultaneously. The analysis was *post hoc*. Many of the results are not reported since they do not add anything, e.g. using STAR on reduced graphs with centroid training data. Had the experimentation been designed better initially, these unnecessary runs could have been avoided. This would have freed up time for investigations which are now consigned to 'further work'.
- A more principled approach would also have identified that the data initially supplied by GSK were not as supposed. Some compounds were duplicated and others were indistinguishable on the basis of the original set of features (properties). This is a potential pitfall in any applied research. In hindsight, the maxim is: 'always check the data before starting a data mining exercise'.
- On the plus side, this did show unequivocally that the set of fourteen properties had poor discriminatory power.
- The preliminary experiments on these data, which motivated the development of the heuristics, were not well designed. Unequal misclassification costs were used which proved to be undesirable, even for maximizing RA. The maximum number of iterations was set at 5000, which proved too few in many cases. Thus a fair comparison between the original heuristics and the later ones was not possible.
- Before applying the heuristics, it would have been preferable to have evaluated the SVM for a wide range of parameters C and σ. For example, by using the search of LOOMS. I was not aware of the LOOMS software when the analysis was commenced, at the time such an search seemed computationally infeasible. LOOMS was later used by Steven Barrett of GSK to classify these data sets. Analysis of these results should indicate whether the restricted parameter sets affected the performance.
- For the majority of the experiments reported in Chapter 5 the maximum number of iterations was set at 10000. This was to avoid excessively long training times. In some cases the SVM did not converge within this limit. It is not known whether this was due to failure of the stopping criteria, or simply because the particular problems were complex. In any case, a maximum must be set when batch learning several classifiers.
- None of the examples made available were reserved as 'unscreened' compounds for the purposes of a final validation. As the initial analysis used all of the compounds with the 'properties', the results reported here using this representation will be optimistically biased. Since this representation was the worst, this does not seriously affect the conclusions. This fortuity though, does not obviate that not holding out some examples is bad practice in data mining.

- There are a few thousand more compounds for which activity data are available for assay 3. These could be used to validate the results, except for those which used the affinity representation. Unfortunately, this representation was optimal. The screening group at GSK have expressed a willingness to set up assays of off-sample compounds if the results are convincing enough.

- In practice, all pK_i values would not be available for all of the compounds. Using only compounds for which pK_i values were available for a given set of receptors would result in a small and biased data set. A technique which allows missing values in the representation would be preferable. Ideally, the prediction of bioactivity profiles could be treated as a problem of multiple imputation of missing data.

- With regards to the previous point, using pK_i values as descriptors relies on historical screening data. Such data may have been generated under different conditions. Although the pK_i value of a compound for a receptor is theoretically a constant, systematic and random errors will lead to variation from screen to screen.

- The clustering used to select the training data was w.r.t. fingerprints. The compounds were not clustered using the same representations as used to train the classifiers. Also, the fingerprints were not used to train a classifier. The compounds were also clustered at Tanimoto thresholds of 0.90, 0.95, and 0.99, which led to respectively larger training sets (since there are more singletons). These clusterings were not used.

- The clustering was partly inspired by Fung and Mangasarian (2000). In that work k-medioid clustering was used. The clustering being done w.r.t. the same representation as was used for prediction. The methodology used here did not lead to similar conclusions.

- When training data are chosen by clustering they are not i.i.d. as the population. Further, the test data are no longer i.i.d. as the population, since both are chosen from the same finite data set. This is most obvious in the difference in hit rates for the two sets. This could be corrected for. It is not known if the class-conditional densities were also altered. Visualization of the data sets, and support vectors, may guide the data mining process.

Finally, note that the majority of the analysis of the pK_i data is not intended to demonstrate the usefulness of the SVM *per se*. It is intended to demonstrate the usefulness of the various different compound representations, the data mining method, and the SVM heuristics. A thorough evaluation of other techniques, such as decision trees, neural networks, clustering and similarity-based prediction, is required to answer the first point.

6.1.5 HERMES

HERMES was developed as a stopping criterion for the SVM. Empirically, it was shown to reduce the training time without loss in accuracy. The evaluation was restricted to five publicly available data sets with similar characteristics to the pK_i data sets.

- The behaviour of $\varepsilon^l_{\alpha\xi}$ was qualitatively the same as that of the test error during training. However, this was only demonstrated for two data sets. Use of artificial data sets would have allowed exact calculation of the generalization error and better characterization of the behaviour of $\varepsilon^l_{\alpha\xi}$.

- Results throughout the thesis cast doubt on the reliability of $\varepsilon^l_{\alpha\xi}$ for parameter selection. The behaviour of other estimators could be investigated to provide alternative stopping criteria.
- A completely fair comparison of the reduction in the number of iterations due to HERMES is only possible if the same QP solver is used. The Matlab implementation with HERMES used Hildreth and D'Espo, whereas BSVM uses TRON.
- An implementation of the SVM with HERMES in C or C++ would allow a comparison of training times with state-of-the-art packages such as BSVM and SVM^{light}.
- The persistence was set at 100. This worked well, but was completely arbitrary. Extensive characterization on artificial data sets could suggest a rule-of-thumb for this parameter, and reveal how sensitive the performance is to its value.

6.1.6 LAIKA

LAIKA was developed as a heuristic method for automatically tuning the width σ of a Gaussian kernel for SVM classification. Some comments on the development and application of LAIKA are as follows.

- LAIKA was developed on two public domain data sets frequently used in the evaluation of SVMs. It was evaluated on one of these and four other data sets, on which we would expect the intial heuristic to lead to overfitting.
- Visualization of the data sets may have better illustrated the motivation behind LAIKA.
- As noted above, a more extensive evaluation on artificial data sets, would have led to a fuller characterization of its behaviour.
- The model selection time was not reduced below that of some other methods for tuning σ, as was intended.
- Jaakkola et al. (2000) define σ_{JDH} with the positive and negative points transposed from the definition used here. This was not initially realized. Tommi Jaakkola[2] states that this is important when there are few positives. Fortunately, this wasn't the case for any of the data sets on which LAIKA or σ_{JDH} were evaluated.
- Burbidge (2002a) reported good performance and a considerable decrease in training time for an early version of LAIKA. However, v1.0 here, which is similar, had the longest training times. The version used in Burbidge (2002a) was affected by the same bugs that affected STAR in Burbidge et al. (2001b), i.e. the algorithm terminated too early. That this only deteriorated the performance on Diabetes suggests that a looser stopping criterion for LAIKA v2.2 could be used.
- The effect of using HERMES as a stopping criterion for LAIKA was not investigated in any detail. It was shown to be useful for only two data sets, and one value of C.
- Again, the choice of a persistence of 100 was arbitrary, other values were not tried.
- The initial value was taken as σ_{JDH} for all experiments. Anecdotal evidence suggests that starting from $\sigma_0 = 1$ or $\sigma_0 = \sqrt{d/2}$ leads to similar final estimates. This has not been systematically investigated.

[2]Personal communication.

- The choice $h_a \in \{20, 40, \ldots, 100\}$ was arbitrary.
- In Chapter 4, it seemed that LAIKA tuned σ to compensate for a poor choice of C. For those data sets a low C was optimal. A higher C gives less explicit regularization; LAIKA increased σ, thus increasing the implicit regularization. In Chapter 5, it seemed that a higher C was *required*, since a low C had a knock-on effect which caused LAIKA to over-regularization. The interplay and causality between these has not been further investigated.

6.1.7 STAR

The heuristic STAR was developed to reduce the number of support vectors (SVs) in the SVM solution. Some comments on the heuristic are as follows.

- There are many SVs in the solutions obtained when classifying noisy and complex data sets such as the pK_i data.
- STAR reduced the number of SVs as expected.
- The SVM was trained to a global optimum of the QP for the evaluation in Chapter 4. The algorithm failed to converge after 100000 iterations on Adult4 with $C = 1000$.
- HERMES was used as the stopping criterion on the pK_i data. However, no evaluation of this criterion with STAR was made beforehand.
- As for the other heuristics, a more exhaustive evaluation on artificial data sets with various characteristics, and for other kernels and other parameter settings, would have provided a better characterization of the behaviour of STAR.
- The update period, $h_e = 100$, was arbitrary. A characterization of the heuristic could lead to a rule-of-thumb.

Since the development of STAR, there have been a number of other attempts to reduce the number of SVs. Schölkopf et al. (1998) solve a quadratic program to find a sparser representation of the SVM solution. This takes roughly as long again as training the SVM. Schohn and Cohn (2000) use a heuristic active learning approach to produce a sparse solution and compare this to a method similar to RaR (Burbidge et al., 2001b). Downs et al. (2001) describe a method for identifying a subset of the support vectors which are linearly independent, the weights α_i for these can be recalculated such that the hyperplane is expressible in terms of this subset only. A comparison between STAR and these three techniques is left as further work.

6.2 Further Work

The critical assessment in the previous section obviously suggests many avenues for further work. Any point of the form 'it would have been preferable to do X' can be transformed into 'further work on this should include X'. Some of these statements can be made for the majority of the analyses presented in this thesis. These are summarized in Section 6.2.1 to avoid repetition. The remainder of the suggestions for further work falls into two categories: specific research on a topic already addressed, and general ideas for new data mining projects and SVM heuristics. Most of the former type have already been mentioned, either in the respective chapter summaries or as suggested by the criticisms in the previous section. These are collected together in Section 6.2.2. The latter type are fewer and require more explanation. These are sketched out in Section 6.2.3.

6.2.1 General Comments

The drug discovery problems in this thesis have been formulated as classification problems. As noted above, this was not a good representation for the QSAR data. The dependent variable for all of the data sets is a real-valued activity. It is common practive, especially for the noisier HTS data, to threshold the activity and label compounds as 'active' or 'inactive'. An immediate avenue for further work is to ignore these thresholds and attempt to predict the real-valued activity. That is, all of the three problems could be treated as regression problems. The SVM for regression has been shown to be competitive with other learning algorithms in a variety of domains (Mangasarian and Musicant, 1999; Smola and Schölkopf, 1997; Vapnik et al., 1997). The problems could also be treated as ones of ordinal regression. Recently, the SVM has recently been adapted to solving this directly (Herbrich et al., 2000).

It is of great interest to GSK to know which representations of a compound are most useful for prediction. Ideally, it is desired to know the best feature set for prediction of affinity to a given receptor, a given group of receptors, and to receptors in general. Three feature sets were used for the HTS analysis and four representations for the pK_i analysis. No definite conclusions can be drawn the HTS data. Further work could involve feature selection within the feature sets, including those not used. On the pK_i data the affinity fingerprint was found to be optimal. Feature selection for each receptor would highlight relationships between the receptors.

An alternative to feature selection is model combination. When more than one classifier has been learned for a given target, for example based on different representations, the learned classifiers can be combined. There are numerous methods for combining classifiers that could be used to aggregate the SVM solutions already found during this work (Scott et al., 1998; Roli and Kittler, 2002).

It is not possible to assess every technique on every problem. However, in Chapter 2, the techniques of clustering and similarity-based prediction were described. It would be useful to apply these to the drug discovery data sets in order to quantify the relative performance of the standard SVM. These techniqes do not seem to be much used in the field of supervised learning. They may exhibit good performance for other applications. Neural networks are also popular in QSAR. On the HTS and pK_i data, only automatically tuned 'dynamic' nets were used. These performed poorly. A fuller evaluation of other algorithms should highlight the benefits of the SVM re parameter selection.

Regarding the SVM, there are numerous further experiments that could be run on all of the data sets. The number of values of C and σ used was fairly small. Now that LOOMS is available, it is possible to perform a search over parameter space, at least for a Gaussian kernel. (The authors are developing the software to handle other kernels.) Very little work has been presented here on the problem of choosing C, or on assessing the sensitivity of the SVM to its value. The limited results suggest that the estimator $\varepsilon^l_{\alpha\xi}$ is not reliable for selection of C. Other error estimators could be used, in particular, the VC bound when errors are penalized quadratically.

Only linear and Gaussian kernels have been used. These are very popular but there are numerous other possibilities (Genton, 2001). Polynomial kernels and the sigmoid kernel were initially used during exploratory data analysis and were found to suffer from very long training times.

Further ideas for the development of the heuristics are given below. A general point for all of the heuristics is the following. The heuristics all introduce an extra parameter, viz. the number of iterations before the heuristic criterion is checked. The effect of this on STAR was investigated briefly by Burbidge et al. (2001b). The effect on LAIKA was investigated in Chapter 4. For all of the heuristics, a thorough characterization should involve evaluation on artificial data sets.

Variables of interest include:

- l, the number of training examples;
- d, the dimensionality;
- the signal-noise ratio;
- q, the working set size;
- the kernel and its parameters; and,
- C, the regularization constant.

Ideally, a rule-of-thumb would be developed, with the SVM being fairly insensitive to the value calculated.

6.2.2 Specific Ideas

In this section, various suggestions for further work on each of the data sets and heuristics are collected.

6.2.2.1 QSAR

The QSAR data were analysed in Chapters 2 (Section 2.4.1) and 3 (Section 3.3.1). Some avenues for further work include the following.

- Use the support vectors as centres for a Gaussian RBF network. This should show whether the improved performance of the SVM was due to the location of the centers (Schölkopf et al., 1997b).
- Use a sigmoid kernel for the SVM. An improved performance over the Gaussian kernel would suggest that the sigmoid is a better model for the data than the Gaussian.
- The representation was six blocks of nine features: one block for each of three substitution sites for each of the two compounds. This information is lost when using an off-the-shelf kernel. An alternative would be to design a kernel that incorporates this additional knowledge.
- The classifications obtained could be converted to a ranking in order to compare the performance with the ILP approach (King et al., 1992).

6.2.2.2 HTS

The HTS data were analysed in Chapters 2 (Section 2.4.2) and 3 (Section 3.3.3). Further work on this problem could include the following.

- A potential application of this work is to identifying false positives and negatives in the HTS screening data. If an accurate and reliable classifier could be learned, then training errors could be flagged as possibly mislabelled. In order to validate such an approach, compounds would have to be screened at least in duplicate. Such data have been generated at GSK.
- The two best techniques for this problem were logistic regression and the SVM. This suggests combining the two, i.e. regularized kernel logistic regression (Platt, 2000; Zhu and Hastie, 2002).

- Isolated hits are not considered progressible. This suggests learning a predictor of the activities of a group of similar compounds. A combination of clustering and prediction is one possibility.
- The results already obtained could be investigated to determine if there is any relationship between the performance, the hit rate, the feature sets, and the nuumber of examples.
- Thresholding the feature weights in the linear models would be a first step towards feature selection. Sensitivity analysis could be used for feature selection with any of the learning machines, including the SVM (Evgeniou et al., 2000).

6.2.2.3 pK_i

The pK_i data were analysed in Chapters 3 (Section 3.3.2) and 5. The analysis of these has been the most extensive. Since a whole chapter has been devoted to the analysis of these data, the future work suggested in the various summaries (Sections 5.1.4, 5.2.3, 5.3.4, and 5.4) is not repeated here. Other possibilities for further analysis include the following.

- The 2048-bit reduced graph (RG) representation should have more discriminative power than the 166-bit MACCS structural keys (Chapter 2, Section 2.2.2). The predictive performance of the SVM was the same with either representation when $C = 1$. A search for the optimal C for each representation is required to confirm or refute the claim.
- The RG and MACCS representations are high-dimensional, sparse and binary. They may thus have similar statistical properties to documents represented as a bag-of-words. Joachims (2001) constructs an idealized document and proves that an SVM classifier with a linear kernel should perform well on such data. Can we analyse these compound representations in a similar manner?
- The Euclidean metric is not the most suitable for fingerprints and structural keys (Chapter 2, Section 2.3.2). The Tanimoto similarity measure could be used instead, i.e. define $K(\mathbf{x}, \mathbf{z}) = \text{Tanimoto}(\mathbf{x}, \mathbf{z})$.
- Although the analysis has been treated as 11 separate classification problems, it could be treated as a single multi-label problem. For example, a chemist may desire compounds that are active against all 5-HT receptors and inactive against all histamine receptors. The precision-recall break-even point (Chapter 3, Section 3.2.1) is often used as a performance measure in this case (Joachims, 1998).

6.2.3 New Directions

Suggestions for further analysis of the heuristics and modifications have been made above and in the summary to Chapter 4. In the following some other heuristics for the SVM are briefly described. These were also motivated in part by the analysis of various pharmaceutical data sets.

6.2.3.1 Transduction

The transductive approach to classification is to perform inductive and deductive steps simultaneously (Vapnik, 1998). We are given a training set and the test set [3] in advance. The aim is to classify the test set with minimum error. Therefore, instead of searching over the space of decision

[3] Also called the *working set* by some authors.

rules, we search over the space of labellings. Note that the predicted labelling may not correspond to the output of any one decision rule in our hypothesis space. If there are k unlabelled examples, then there are 2^k binary classifications. The optimization requires learning an hyperplane for each of the 2^k possible labellings. The transductive SVM solution is the labelling of the test set that leads to the largest margin. This search is combinatorial, a heuristic search is given by Joachims (1999b). Mathematical programming techniques have also been applied to the problem (Fung and Mangasarian, 2000; Bennett, 1999).

An alternative is to cluster the test data, as suggested by Vapnik (1998), and perform a heuristic search over labellings of the clusters, possibly allowing points to move cluster if they are repeatedly mislabelled during the search. Further, only test points classified with low confidence by the inductive SVM need be included in the search. This approach was presented at NIPS'99 (Burbidge, 1999), but not investigated further.

The advantage of the transductive approach is that the decision function is only learned in the region of interest. This may be beneficial in drug discovery. A group of unscreened compounds may be of interest for different reasons than those already screened and may not be in the same region of chemical space. To attempt to learn a classifier for the whole of chemical space from a sample drawn from one region of it is unreasonable. To learn a classifier for a subregion is a modest aim. Strictly, the theory behind the transductive approach only holds if the test data are i.i.d. as the test data. Yet, it seems as though transduction would be most useful when this is not the case (Klinkenberg and Joachims, 2000).

6.2.3.2 Imputation of Missing Data

Multiple missing attribute values can be imputed independently or collectively. When there are many complete data records it is feasible to learn a predictive model for each feature. If most of the data records have some entries missing this is not feasible. In this case one desires to learn multiple imputations simultaneously. One approach is to train a neural network with multiple outputs and a network structure that maintains consistency in the imputation. Neural networks can be difficult to tune and slow to train. The SVM has only a single output and cannot be used directly for multiple imputation. A committe of SVMs could be trained that can perform multiple imputation. Starting from an initial random guess of all of the missing values, one SVM is trained to predict each feature, based on the other features. The collective predictions are then used as a first approximation for the missing values and the process is repeated until a stable set of predictions is reached for the missing values.

With binary SVM classifiers, this approach is only possible when all of the features are binary. This is case for the pK_i data if the features are taken not be the SQ pK_i values, but these affinities thresholded as 'active' or 'inactive'. For multi-valued features, a network of SVM regressors could be used. The pK_i data inspired this idea, since in practice, there are many compounds that have been screened against some targets, but few that have been screened against all. The problem of predicting binding affinity from binding affinity is therefore more naturally framed as one of multiple imputation of missing data, rather than a straightforward classification (or regression) problem.

When predicting the missing values, the examples on which we wish to predict are known. This step could be treated as one of transductive inference.

6.2.3.3 Maximixing Relative Advantage

In Chapter 3 (Section 3.2.2), the following was proposed as a first order approximation to maximizing the relative advantage (RA):

$$\underset{\mathbf{w},b}{\text{Minimize}} \quad \frac{FP}{N} + \lambda \frac{FN}{P},$$

where $\lambda = FP/N$ is unknown. This risk functional has been used with $\lambda = 1$, which errs in favour of correctly classifying positive points (e.g. active compounds). An alternative is an iterative approach. Starting with $\lambda_0 = 0.5$, say, solve the optimization. Calculate $\lambda_1 = FP_0/N$, where FP_0 is the number of mislabelled negative training examples. Iterate until the change in λ falls below some tolerance. To speed this up, λ could be updated after every, say, 100 iterations, as for the heuristics described in Chapter 4.

For the SVM, changing λ amounts to changing the ratio of the misclassification costs, which is equivalent to changing the ratio of the regularization parameters C^+, C^-. Retraining of the SVM with a new λ should therefore be fairly rapid. The iterative maximization could be applied to any classification algorithm which minimizes a risk.

6.2.3.4 Aggregating SVMs

The training time of the SVM scales empirically as $O(l^2)$ (Platt, 1999; Joachims, 1999a). This can become computationally infeasible for very large data sets. Bradley and Mangasarian (2000) have proposed linear programming methods for fast training on massive data sets. Another possibility is to segment the training data, either randomly or by clustering, and train an SVM for each segment of data. The learned classifiers must then be combined in some way. Evgeniou (2000) proposes combining the solutions by learning a linear relationship between the individual outputs and the labels of the training data. An alternative is to treat the SVMs as preprocessors. That is, the support vectors (SVs) within each segment are considered as candidate SVs for the SVM trained on the whole data. Points which are not SVs of the SVMs trained on the segments are discarded. A single SVM can then be trained on the subset of points that are SVs from the individual SVMs. Other refinements include discarding SVs that are training errors (cf. STAR), and partitioning the data several times, and retaining only those points which are SVs for the majority of partitions.

6.3 Conclusions

The scientific contributions presented in this thesis are to the field of data mining. A number of general points have been raised and methodologies presented and evaluated. The chief contributions are to the domain of support vector classification. These are the three heuristics HERMES, LAIKA and STAR. To the respective authors' knowledge, these heuristics have not yet been used in the SVM community. Work introducing these algorithms has been published or accepted for publication. In particular, the stopping criterion HERMES has been accepted for publication by C.-W. Hsu, one of the coauthors of BSVM (Hsu and Lin, 2002) which is among the fastest SVM implementations. T. Joachims[4] has also expressed an interest in incorporating the heuristic into his state-of-the-art SVM package SVM^{light} (Joachims, 1999a). The heuristic STAR has also received interest from the authors of the first SVM book Cristianini and Shawe-Taylor (2000)[5].

[4]Personal communication.
[5]Chris Watkins, personal communication

The application of these heuristics to problems in drug discovery was broadly successful. Each of the algorithms achieved the aims for which it was designed to some extent. The shortcomings and possible solutions have been discussed. It should be noted that the aim of GlaxoSmithKline in providing the various data sets was to find out if anything at all could be said. The criterion for success was essentially: Can we do better than random in predicting activity? The answer provided by the research presented here is yes, with characterizations of the heuristics and the data sets and proposals for use in practice.

This leads to the question of: What next? Ideally, I would like to take what I have learned about data analysis, in particular classification, and formulate a well-defined, statistically rigorous set of experiments to validate the claims made. In practice, this is impossible, as the commercial environment is not a purely research one. The numerous data sets and problems tackled have well-prepared me for aiming for a second best. Approaching industrial and commercial problems with the mindset of a pragmatic theorist is an almost oxymoronic balance, but one that must be held as an ideal to ensure that the results obtained can be meaningfully interpreted. A balance must also be found between exploratory work and the more rigorous experiments such work suggests and the actual implementation of the techniques developed. As noted in the introduction, there is not much point doing applied science unless it is applied.

Bibliography

Abraham, M., 2001. Prediction of physical chemical and biological processes from structure using LFER methodology. Presentation.

Aizerman, M., Braverman, E., Rozonoer, L., 1964. Theoretical foundations of the potential function method in pattern recognition learning. Automations and Remote Control 25, 821–837.

Arabie, P., Hubert, L., Soete, G. (Eds.), 1996. Clustering and Classification. World Scientific, Singapore.

Arunlakshana, O., Schild, H., 1959. Some quantitative uses of drug antagonists. British Journal of Pharmacology 14, 48–58.

Ash, R., 1990. Information Theory. Dover.

Bellman, R., 1961. Adaptive Control Processes: A Guided Tour. Princeton University Press, NJ.

Bennett, K., 1999. Mathematical programming methods for support vector machines. In: Schölkopf et al. (1999a), pp. 293–326.

Bennett, K., Mangasarian, O., 1992. Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods and Software 1, 23–34.

Bevan, C., Hill, A., Reynolds, D., Valko, K., 2001. Rapid physicochemical profiling. Presentation.

Bishop, C., 1995. Neural Networks for Pattern Recognition. Clarendon Press.

Blake, C. L., Merz, C. J., 1998. UCI repository of machine learning databases . URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Böhm, H.-J., Schneider, G. (Eds.), 2000. Virtual Screening for Bioactive Molecules. Vol. 10 of Methods and Principles in Medicinal Chemistry. Wiley-VCH, Weinheim, Germany.

Borman, S., 1990. New QSAR techniques eyed for environmental assessments. Chemical Engineering News 68, 20–23.

Bradley, P., Mangasarian, O., 2000. Massive data discrimination via linear support vector machines. Optimization Methods and Software 13 (1), 1–10.

Bravi, G., Gancia, E., Green, D., Hann, M., 2000. Virtual Screening for Bioactive Molecules, Ch. Modelling Structure-Activity Relationships. Vol. 10 of Böhm and Schneider (2000), pp. 81–116.

Breiman, L., 1994. Bagging predictors. Tech. Rep. 421, Department of Statistics, University of California, Berkeley, CA.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth International Group, Belmont, CA.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., M. Ares, J., Haussler, D., 2000. Knowledge-based analysis of microarray gene expression data using support vector machines. Proceedings of the National Academy of Sciences 97 (1), 262–267.

Brown, R., Martin, Y., 1996. Use of structure-activity data to compare structure-based clustering methods and descriptors for use in compound selection. Journal of Chemical Information and Computer Science 36, 572–584.

Brown, R., Martin, Y., 1997. The information content of 2D and 3D structural descriptors relevant to ligand-receptor binding. Journal of Chemical Information and Computer Science 37, 1–9.

Burbidge, R., December 1999. Making transductive classification feasible. Presentation. NIPS'99 Workshop on Using Unlabelled Data for Supervised Learning. Breckenridge, Co.
URL `http://stats.ma.ic.ac.uk/ rdb/presentations/nips99-transduction.pps`

Burbidge, R., March 2002a. Adaptive kernels for support vector classification. In: Proceedings of the MSRI Workshop on Non-Linear Estimation and Classification. Lecture Notes in Computer Science. Springer-Verlag, Berkeley, Ca., to appear.

Burbidge, R., February 2002b. Stopping criteria for SVMs. Tech. rep., Statistics Section, Imperial College, 180 Queen's Gate, London, SW7 2BZ, submitted to Special Session on Support Vector Machines and Kernel Methods, ICONIP'02.

Burbidge, R., Buxton, B., March 2001. An introduction to support vector machines for data mining. In: Sheppee, M. (Ed.), Keynote Papers, Young OR12. Operational Research Society, Operational Research Society, University of Nottingham, pp. 3–15.

Burbidge, R., Trotter, M., B.Buxton, S.Holden, 2000. Drug design by machine learning: support vector machines for pharmaceutical data analysis. In: Martin, A., Corne, D. (Eds.), Proceedings of the AISB'00 Symposium on Artificial Intelligence in Bioinformatics symposium, AISB00, 17^{th}–20^{th} April, 2000. University of Birmingham, pp. 1–4.

Burbidge, R., Trotter, M., B.Buxton, S.Holden, December 2001a. Drug design by machine learning: support vector machines for pharmaceutical data analysis. Computers and Chemistry 26 (1), 4–15.

Burbidge, R., Trotter, M., B.Buxton, S.Holden, 2001b. STAR — sparsity through automated rejection. In: Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence: 6th International Work-Conference On Artificial and Natural Neural Networks, IWANN 2001, Proceedings, Part 1. Granada, Spain, June 2001, pp. 653–660.

Burges, C., Crisp, D., 2000. Uniqueness of the SVM solution. In: Solla et al. (2000), pp. 223–229.

Burges, C. J. C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2 (2), 1–47.

Butina, D., 1999. Unsupervised data base clustering based on Daylight's fingerprint and Tanimoto similarity: a fast and automated way to cluster small and large data sets. Journal of Chemical Information and Computer Science 39, 747–750.

Chapelle, O., Vapnik, V., 2000. Model selection for support vector machines. In: Solla et al. (2000), pp. 230–236.

Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., 2002. Choosing multiple parameters for support vector machines. Machine Learning 46, 131–160.

Cheng, Y.-C., Prusoff, W., 1973. Relationship between the inhibition constant (k_i) and the concentration of inhibitor which causes 50 per cent inhibition ($i50$) of an enzymatic reaction. Biochemical Pharmacology 22, 3099–3108.

Clementine 6.0, 2001. SPSS Inc. Headquarters, 233 S. Wacker Drive, 11th floor, Chicago, Illinois 60606.

Cortes, C., Vapnik, V., 1995. Support vector networks. Machine Learning 20, 273–297.

Cristianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press.

Cristianini, N., Shawe-Taylor, J., Campbell, C., 1998. Dynamically adapting kernels in support vector machines. In: Kearns, M., Solla, S., Cohn, D. (Eds.), Advances in Neural Information Processing Systems, 11. The MIT Press, Denver, CO.

Curran, C., April 2001. Novartis tackles rise in respiratory diseases. Chemistry and Industry 7, 205.

Devijer, P., Kittler, J., 1982. Pattern Recognition: A Statistical Approach. Prentice-Hall.

Dixon, S., Villar, H., 1998. Bioactive diversity and screening library selection via affinity fingerprinting. Journal of Chemical Information and Computer Science 38, 1192–1203.

Downs, T., Gates, K., Masters, A., 2001. Exact simplification of support vector machine solutions. Journal of Machine Learning Research 2, 293–297.

Drucker, H., Schapire, R., Simard., P., 1993. Improving performance in neural networks using a boosting algorithm. In: Hanson, S., Cowan, J., Giles, C. (Eds.), Neural Information Processing Systems 5. Morgan Kaufmann, Denver, C0, pp. 42–49.

Duan, K., Keerthi, S., Poo, A., 2001. Evaluation of simple performance measures for tuning SVM hyper parameters. Tech. Rep. CD-01-11, Department of Mechanical Engineering, National University of Singapore, 10, Kent Ridge Crescent, 119260, Singapore.

Duda, R., Hart, P., 1973. Pattern Classification and Scene Analysis. John Wiley & Sons.

Eglen, R., Schneider, G., Böhm, H.-J., 2000. High-throughput screening and virtual screening: Entry points to drug discovery. In: Böhm and Schneider (2000), pp. 1–14.

Eriksson, L., Johansson, E., 1996. Multivariate design and modeling in QSAR. Chemometrics and Intelligent Laboratory Systems 34, 1–19.

Evgeniou, T., June 2000. Learning with kernel machine architectures. Ph.D. thesis, MIT.

Evgeniou, T., Pontil, M., Papageorgiou, C., Poggio, T., 2000. Image representations for object detection using kernel classifiers. In: Proceedings of ACCV2000 The Fourth Asian Conference on Computer Vision January 8th–11th 2000. Lecture Notes in Computer Science. Springer, The Grand Hotel, Taipei, Taiwan, to appear.

Fellenz, W., June 2001. Reduced support vector selection by linear programs. In: Mira and Prieto (2001), pp. 677–684.

Feng, J., June 2001. Non-symmetric support vector machines. In: Mira and Prieto (2001), pp. 418–426.

Freund, Y., Schapire, R., 1999. Large margin classification using the perceptron algorithm. Machine Learning 37 (3), 277–296.

Friedman, J., 1997. On bias, variance, 0/1—loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery 1, 55–77.

Frieß T.-T., Cristianini, N., Campbell, C., 1998. The kernel-adatron: a fast and simple learning procedure for support vector machines. In: Shavlik, J. (Ed.), Machine Learning Proceedings of the Fifteenth International Conference (ICML '98). Morgan Kaufmann, San Francisco, CA, pp. 188–196.

Fukunaga, K., 1972. Introduction to Statistical Pattern Recognition, 1st Edition. Academic Press.

Fung, G., Mangasarian, O., October 1999. Semi-supervised support vector machines for unlabelled data classification. Tech. Rep. 99-05, Data Mining Institute, Computer Sciences Department, University of Wisconsin.

Fung, G., Mangasarian, O., 2000. Data selection for support vector machine classifiers. In: Ramakrishnan, R., Stolfo, S. (Eds.), KDD-2000, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 20-23, 2000, Boston, MA. ACM, pp. 20–23.

Gentile, C., 2001. A new approximate maximal margin classification algorithm. In: Leen, T., Dietterich, T., Tresp, V. (Eds.), Advances in Neural Information Processing Systems 13. The MIT Press, Denver, CO.

Genton, M., 2001. Classes of kernels for machine learning: A statistics perspective. Journal of Machine Learning Research 2, 299–312.

Gillet, V., Downs, G., Holliday, J., Lynch, M., Dethlefsen, W., 1991. Computer storage and retrieval of generic chemical structures in patents. 13. reduced graph generation. Journal of Chemical Information and Computer Science 31, 260–270.

Gillet, V., Willett, P., Bradshaw, J., 1998. Identification of biological activity profiles using substructural analysis and genetic algorithms. Journal of Chemical Informatics and Computer Science 38, 165–179.

Gillet, V., Willett, P., Bradshaw, J., 28^{th}–29^{th} October 1999. Reduced graphs as descriptors of bioactivity. Presentation.
URL `http://www.daylight.com/meetings/emug99/Gillet/`

Guénoche, A., Hansen, P., Jaumard, B., 1991. Efficient algorithms for divisive hierarchical clustering. Journal of Classification 8, 5–30.

Hammett, L., 1970. Physical Organic Chemistry, 2nd Edition. McGraw-Hill, New York.

Hand, D., Till, R., 2001. A simple generalisation of the area under the roc curve for multiple class classification problems. Machine Learning 45, 171–186.

Hanley, J., McNeil, B., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29–36.

Hann, M., Green, R., 1999. Chemoinformatics — a new name for an old problem? Current Opinion in Chemical Biology 3, 379–383.

Hansch, C., 1969. A quantitative approach to biochemical structure-activity relationships. Acct. Chem. Res. 2, 232–239.

Hansch, C., Leo, A., 1995. Exploring QSAR Fundamentals and Applications in Chemistry and Biology. Vol. 1. American Chemical Society, Washington, D.C.

Herbrich, R., Graepel, T., Obermayer, K., 2000. Large margin rank boundaries for ordinal regression. In: Smola et al. (2000), pp. 115–132.

Herbrich, R., Graepel, T., Williamson, R., 2002. The structure of version space, personal communication.

Hertz, J., Krogh, A., Palmer, R., 1991. Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, CA.

Hijfte, L. V., Marciniak, G., Froloff, N., 1999. Combinatorial chemistry, automation and molecular diversity: new trends in the pharmaceutical industry. Journal of Chromatography B 725, 3–15.

Hildreth, C., 1957. A quadratic programming procedure. Naval Research Logistics Quarterly 4, 79–85.

Hollinger, M., 1997. Inroduction to Pharmacology. Taylor & Francis, Washington, D.C.

Howard, A., McAllister, G., Feighner, A., Liu, Q., Nargund, R., der Ploeg, L. V., Patchett, A., March 2001. Orphan G-protein-coupled receptors and natural ligand discovery. TRENDS in Pharmacological Sciences 22 (3), 132–140.

Hsu, C.-W., Lin, C.-J., 2002. A simple decomposition method for support vector machines. Machine Learning 46, 291–314.

Huber, P., 1981. Robust Statistics. John Wiley & Sons.

Jaakkola, T., Diekhans, M., Haussler, D., 1999. Using the fisher kernel method to detect remote protein homologies. In: Lengauer, T., Schneider, R., Bork, P., Brutlag, D., Glasgow, J., Mewes, H.-W., Zimmer, R. (Eds.), Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology. AAAI Press, Heidelberg, Germany, pp. 149–158.

Jaakkola, T., Diekhans, M., Haussler, D., 2000. A discriminative framework for detecting remote protein homologies. Journal of Computational Biology 7, 95–114.

James, C., Weininger, D., Delaney, J., July 2000. Daylight Theory Manual Daylight 4.71. Daylight Chemical Information Systems, Inc., 27401 Los Altos, Suite #360, Mission Viejo, CA 92691.

Jarvis, E., Patrick, E., 1973. Clustering using a similarity measure based on shared nearest neighbours. IEEE Transactions on Computing C 22, 1025–1034.

Joachims, T., April 1998. Text categorization with support vector machines: learning with many relevant features. In: Nedellec, C., Rouveirol, C. (Eds.), Machine Learning: ECML-98, 10th European Conference on Machine Learning. Vol. 1398 of Lecture Notes in Artificial Intelligence. Springer, Chemnitz, Germany.

Joachims, T., 1999a. Making large-scale SVM learning practical. In: Schölkopf et al. (1999a), pp. 169–184.

Joachims, T., 1999b. Transductive inference for text classification using support vector machines. In: International Conference on Machine Learning.

Joachims, T., 2000. Estimating the generalization performance of a SVM efficiently. In: Langley (2000), pp. 431–438.

Joachims, T., 2001. Support vector machines and document classification. Ph.D. thesis, GMD First, check this.

Karchin, R., Karplus, K., Haussler, D., 2001. Classifying G-protein coupled receptors with support vector machines. Bioinformatics .

Kauvar, L., Higgins, D., Villar, H., Sportsman, J., Engvist-Goldstein, A., Bukar, R., Bauer, K., Dilley, H., Rocke, D., 1995. Predicting ligand binding to proteins by affinity fingerprinting. Chemistry and Biology 2, 107–118.

King, R. D., Muggleton, S., Lewis, R. A., Sternberg, M. J. E., 1992. Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. Proc. Natl. Acad. Sci. USA 89, 11322–11326.

Klinkenberg, R., Joachims, T., 2000. Detecting concept drift with support vector machines. In: Langley (2000).

Kohavi, R., 1996. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In: Simoudis, E., Han, J., Fayyad, U. (Eds.), Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 202–207.

Kövesdi, I., Dominguez, M., Ôrfi, L., Náray-Szabó, G., Varró, A., Papp, J., Mátyus, P., 1999. Application of neural networks in structure-activity relationships. Medicinal Research Reviews 19 (3), 249–269.

Labute, P., 1999. Binary QSAR: A new method for the determination of quantitative structure activity relationships. In: Pacific Symposium on Biocomputing 4. pp. 444–445.

Lang, K., Waibel, A., Hinton, G., 1990. A time delay neural network architecture for isolated word recognition. Neural Networks 3, 33–43.

Langley, P. (Ed.), 2000. Machine Learning Proceedings of the Seventeenth International Conference (ICML '00). Morgan Kaufmann, Stanford, CA.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation 1 (4), 541–551.

Lee, J.-H., Lin, C.-J., November 2000. Automatic model selection for support vector machines. Tech. rep., Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan.

Leo, A., Weininger, D., August 2000. CLOGP Reference Manual. Daylight Chemical Information Systems, Mission Viejo, CA.

Li, Y., Long, P., 2001. The relaxed online maximum margin algorithm. Machine Learning 46, 361–387.

Lin, C.-J., Moré, J., 1999. Newton's method for large-scale bound constrained problems. SIAM Journal of Optimization 9, 1100–1127, software available at `http://www.mcs.anl.gov/~more/tron`.

Lin, Y., Lee, Y., Wahba, G., March 2000. Support vector machines for classification in nonstandard situations. Tech. Rep. 1016, Department of Statistics, University of Wisconsin, 1210, West Dayton St., Madison, WI 53706.

Lipinski, C., Lombardo, F., Dominy, B., Feeney, P., 1997. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Advanced Drug Delivery Reviews 23, 3–25.

Lipnick, R., 1986. Charles Earnest Overton: Narcosis studies and a contribution to general pharmacology. Trends in Pharmacological Science 7, 161–164.

Littlestone, N., Warmuth, M., 1986. Relating data compression and learnability. Tech. rep., University of California, Santa Cruz.

Lunts, A., Brailovskiy, V., 1967. Evaluation of attributes obtained in statistical decision rules. Engineering Cybernetics 3, 98–109.

MACCS-II, 1994. MACCS-II Menu Reference Manual version 2.2. MDL Information Systems, San Leandro, CA.

Manallack, D., Livingstone, D., 1999. Neural networks in drug discovery: have they lived up to their promise? Eur. J. Med. Chem. 34, 95–208.

Mangasarian, O., 2000. Generalized support vector machines. In: Smola et al. (2000), pp. 135–146.

Mangasarian, O., Musicant, D., November 1999. Robust linear and support vector regression. Tech. Rep. 99-09, Data Mining Institute, Computer Sciences Department, University of Wisconsin.

Mangasarian, O., Musicant, D., June 2000. Lagrangian support vector machines. Tech. Rep. 00-06, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.

Mangasarian, O., Wolberg, W., September 1990. Cancer diagnosis via linear programming. SIAM News 23 (5), 1&18.

Matlab 6.0, 2001. The Mathworks, Inc., 3, Apple Hill Drive, Natick, MA 01760-2098.

McGregor, M., Pallai, P., 1997. Clustering of large databases of compounds: using the MDL "keys" as structural descriptors. Journal of Chemical Information and Computer Science 37, 443–448.

McLoone, S., Irwin, G., 2001. Improving neural network training solutions using regularisation. Neurocomputing 37, 71–90.

Menard, P., Lewis, R., Mason, J., 1998. Rational screening set design and compound selection: cascaded clustering. Journal of Chemical Information and Computer Science 38, 497–505.

Mercer, J., 1909. Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions Royal Society London A 209, 415–446.

Michie, D., Spiegelhalter, D., Taylor, C., 1994. Machine Learning, Neural and Statistical Classification. Prentice Hall, Englewood Cliffs, N.J., data available at anonymous ftp: `ftp.ncc.up.pt/pub/statlog/`.

Mira, J., Prieto, A. (Eds.), June 2001. Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence: 6th International Work-Conference On Artificial and Natural Neural Networks, IWANN 2001, Proceedings, Part 1. Vol. 2084 of Lecture Notes in Computer Science. Springer-Verlag, Granada, Spain.

Mitchell, T., 1997. Machine Learning. McGraw-Hill International.

Morris, J., Bruneau, P., 2000. Prediction of physicochemical properties. In: Böhm and Schneider (2000), pp. 33–58.

Muggleton, S., Bryant, C., Srinivasan, A., May/June 2000. Measuring performance when positives are rare: Relative advantage versus predictive accuracy — a biological case-study. In: de Mantaras, R. L., E.Plaza (Eds.), Machine Learning: ECML 2000 11th European Conference on Machine Learning. Vol. 1810 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Barcelona, Catalonia, Spain.

Murthy, S. K., Kasif, S., Salzberg, S., 1994. A system for oblique induction of decision trees. Journal of Artificial Intelligence Research 2, 1–32.

Neal, R., 1996. Bayesian Learning for Neural Networks. Springer.

Ng, A., 1997. Preventing overfitting of cross-validation data. In: Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann, pp. 245–253.

Osuna, E., Freund, R., Girosi, F., 1997a. An improved training algorithm for support vector machines. In: Principe, J., Giles, L., Morgan, N., Wilson, E. (Eds.), Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing. Amelia Island, FL., pp. 276–285.

Osuna, E., Freund, R., Girosi, F., May 1997b. Support vector machines: training and applications. AI Memo 1602, MIT.

Platt, J., 1999. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf et al. (1999a), pp. 185–208.

Platt, J., 2000. Probabilities for SV machines. In: Smola et al. (2000), pp. 61–74.

Pomerleau, D., 1993. Knowledge-based training of artificial neural networks for autonomous robot driving. In: Connell, J., Mahadevan, S. (Eds.), Robot Learning. Kluwer Academic, pp. 19–43.

Popper, K., 1968. The Logic of Scientific Discovery, 2nd Edition. Harper Torch Books, New York.

Provost, F., Fawcett, T., 2000. Robust classification systems for imprecise environments. Machine Learning To appear.

QSAR, U., Group, C., 2001. Spring meeting.

Quinlan, J., 1992. C4.5: Programs for Machine Learning. Morgan Kaufmann.

Quinlan, R., 1987. Simplifying decision trees. International Journal of Man-Machine Studies 27, 221–234.

Rätsch, G., Onoda, T., Müller, K.-R., 2001. Soft margins for AdaBoost. Machine Learning 42 (3), 287–320.

Ripley, B., 1996. Pattern Recognition and Neural Networks. Cambridge University Press.

Roli, F., Kittler, J. (Eds.), 2002. Multiple Classifier Systems, Third International Workshop, MCS 2002, Cagliari, Italy, June 24-26, 2002. Proceedings. Lecture Notes in Computer Science. Springer-Verlag.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65, 386–408.

Sadowski, J., 2000. Optimization of chemical libraries by neural networks. Current Opinion in Chemical Biology 4, 280–282.

Schneider, G., 2000. Neural networks are useful tools for drug design. Neural Networks 13, 15–16.

Schohn, G., Cohn, D., 2000. Less is more: Active learning with support vector machines,. In: Langley (2000).

Schölkopf, B., Burges, C., Smola, A. (Eds.), 1999a. Advances in Kernel Methods: Support Vector Learning. The MIT Press.

Schölkopf, B., Knirsch, P., Smola, A., Burges, C., 1998. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In: DAGM Symposium Mustererkennung. LNCS. Springer-Verlag.

Schölkopf, B., Shawe-Taylor, J., Smola, A., Williamson, R., 1999b. Kernel-dependent support vector error bounds. In: Ninth International Conference on Artificial Neural Networks. Institute of Electrical Engineers, University of Edinburgh, UK, 7^{th}–10^{th} September, 1999, pp. 304–309.

Schölkopf, B., Smola, A., Müller, K.-R., 1997a. Kernel principal components analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (Eds.), Artificial Neural Networks — ICANN'97. Vol. 1327 of Lecture Notes in Computer Science. Springer, pp. 583–588.

Schölkopf, B., Sung, K.-K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V., 1997b. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing 45 (11).

Scott, M., Niranjan, M., Prager, R., May 1998. Parcel: feature subset selection in variable cost domains. Tech. Rep. CUED/F-INFENG/TR.323, Cambridge University Engineering Department.

Seydel, J., 1966. Prediction of *in vitro* activity of sulfonamides, using Hammett constants or spectrophotometric data of the basic amines for calculation. Molecular Pharmacology 2, 259–265.

Shawe-Taylor, J., Bartlett, P., Williamson, R., Anthony, M., October 1996. Structural risk minimization over data-dependent hierarchies. Tech. Rep. NC-TR-96053, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK.

Sigillito, V., Wing, S., Hutton, L., Baker, K., 1989. Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest 10, 262–266.

Smith, H., Draper, N., 1998. Applied Regression Analysis, 2nd Edition. John Wiley & Sons.

Smith, J., Everhart, J., Dickson, W., Knowler, W., Johannes, R., 1998. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Greenes, R. (Ed.), Proceedings of the Symposium on Computer Applications in Medical Care. Los Alamitos, CA: IEEE Computer Society Press, Washington, pp. 261–265.

Smola, A., Bartlett, P., Scholkopf, B., Schuurmans, D. (Eds.), 2000. Advances in Large Margin Classifiers. The MIT Press.

Smola, A., Schölkopf, B., 1997. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Tech. Rep. 1064, GMD First, Kekuléstraße 7, 12489 Berlin.

Smola, A., Schölkopf, B., Müller, K.-R., 1998. The connection between regularization operators and support vector kernels. Neural Networks 11, 637–649.

Solla, S., Leen, T., Müller, K.-R. (Eds.), 2000. Advances in Neural Information Processing Systems 12.

Sollich, P., 1999. Probabilistic interpretation and Bayesian methods for support vector machines. In: Ninth International Conference on Artificial Neural Networks. The Institute of Electrical Engineers, London, UK, pp. 91–96.

Sollich, P., 2000. Probabilistic methods for support vector machines. In: Solla et al. (2000), pp. 349–355.

SPSS, 1999. Clementine 5.1.
URL `http://www.spss.com`

Stead, P., 2000. High throughput screening of nature's diversity: successes, issues and future directions. In: Dixon, G., Major, J., Rice, M. (Eds.), High Throughput Screening: The Next Generation. Society of Chemical Industry, BIOS Scientific Publishers Ltd., Oxford, University of Surrey, Guildford, UK, pp. 75–82.

Terstappen, G., Reggiani, A., January 2001. *In silico* research in drug discovery. TRENDS in Pharmacological Sciences 22 (1), 23–26.

Valler, M., Green, D., July 2000. Diversity screening versus focussed screening in drug discovery. Drug Discovery Today 5 (7), 286–293.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag.

Vapnik, V., 1998. Statistical Learning Theory. John Wiley & Sons.

Vapnik, V., Chapelle, O., 2000. Bounds on error expectation for support vector machines. Neural Computation 12 (9), 2013–2036.

Vapnik, V., Chervonenkis, A., 1974. Theory of Pattern Recognition. Nauka, Moscow.

Vapnik, V., Golowich, S., Smola, A., 1997. Support vector method for function approximation, regression estimation and signal processing. In: Mozer, M., Jordan, M., Petsche, T. (Eds.), Neural Information Processing Systems. Vol. 9. The MIT Press, Denver, CO.

Vapnik, V., Lerner, A., 1963. Pattern recognition using generalized portrait method. Automation and Remote Control 24.

Wahba, G., Lin, Y., Zhang, H., 2000. GACV for support vector machines. In: Smola et al. (2000), pp. 297–310.

Walters, W., Stahl, M., Murcko, M., April 1998. Virtual screening — an overview. Drug Discovery Today 3 (4), 160–178.

Ward, J., March 1963. Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association 58 (301), 236–244.

Williamson, R., Smola, A., Schölkopf, B., 1999. Entropy numbers, operators and support vector kernels. In: Schölkopf et al. (1999a), pp. 127–144.

Zhang, J.-H., Chung, T., Oldenburg, K., 1999. A simple statistical parameter for use in evaluation and validation of high throughput screening assays. Journal of Biomolecular Screening 4 (2).

Zhang, J.-H., Chung, T., Oldenburg, K., 2000. Confirmation of primary active substances from high throughput screening of chemical and biological populations: a statistical approach and practical considerations. Journal of Combinatorial Chemistry 2, 258–265.

Zhu, J., Hastie, T., 2002. Kernel logistic regression and the import vector machine. In: Advances in Neural Information Processing Systems 14. The MIT Press, Denver, CO, to appear.

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., Müller, K.-R., 2000. Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics 16 (9), 799–807.